МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
Гомельский государственный технический университет имени П. О. Сухого

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломной работе
на тему: «Программный комплекс управления расписанием
в ГГТУ им. П. О. Сухого»

Разработал студент гр. ИП-42	_____ (подпись)	Прокопенко А. Р. (Ф.И.О.)
Руководитель работы	_____ (подпись)	доцент, к.т.н., Трохова Т. А. (ученое звание, ученая степень, Ф.И.О.)
Консультант по экономической части	_____ (подпись)	доцент, к.э.н., Соловьева Л. Л. (ученое звание, ученая степень, Ф.И.О.)
Консультант по охране труда и технике безопасности	_____ (подпись)	профессор, д.т.н, Кудин В. П. (ученое звание, ученая степень, Ф.И.О.)
Нормоконтроль	_____ (подпись)	Самовендюк Н. В. (ученое звание, ученая степень, Ф.И.О.)
Рецензент	_____ (подпись)	_____ (ученое звание, ученая степень, должность, организация, Ф.И.О.)

Дипломная работа (_____с.) допущена к защите
в Государственной экзаменационной комиссии.

Зав. кафедрой _____ доцент, к.т.н., Трохова Т. А.
(подпись)

Гомель 2023

РЕФЕРАТ

ПРОГРАММНЫЙ КОМПЛЕКС УПРАВЛЕНИЯ РАСПИСАНИЕМ В ГГТУ ИМ. П. О. СУХОГО : дипломная работа / А. Р. Прокопенко. – Гомель : ГГТУ им. П. О. Сухого, 2023. – Дипломная работа: __страницы, __рисунка, __таблиц, __источника, __приложений.

Ключевые слова: _____

Объектом разработки является расписание в ГГТУ им. П. О. Сухого.

Цель работы: разработка программного комплекса управления расписанием в ГГТУ им. П. О. Сухого, позволяющего формировать расписание в автоматическом режиме.

Характеристика проделанной работы: _____
_____
_____

Студент-дипломник подтверждает, что дипломная работа выполнена самостоятельно, приведенный в дипломной работе материал объективно отражает состояние разрабатываемого объекта, пояснительная записка проверена в системе «Антиплагиат» (). Процент оригинальности составляет __ процентов. Все заимствованные из литературных и других источников теоретические и методологические положения и концепции сопровождаются ссылками на источники, указанные в «Списке использованных источников».

РЕЗЮМЕ

Тема работы Программный комплекс управления расписанием в ГГТУ им. П. О. Сухого
Объектом исследования является расписание в ГГТУ им. П. О. Сухого
Цель работы разработка программного комплекса управления расписанием в ГГТУ им. П. О. Сухого, позволяющего формировать расписание в автоматическом режиме
Основным результатом работы является программа, позволяющая составлять расписание в ГГТУ им. П. О. Сухого в ручном и автоматическом режимах

РЭЗЮМЭ

Тэма працы Праграмны комплекс кіравання раскладам у ГГТУ ім. П. О. Сухога
Аб'ектам даследвання з'яўляецца расклад у ГГТУ ім. П. О. Сухога
Мэта працы распрацоўка праграмнага комплексу кіравання раскладам ГГТУ ім. П. О. Сухога, які дазваляе фармаваць расклад у аўтаматычным рэжыме
Асноўным вынікам працы з'яўляецца праграма, якая дазваляе складаць расклад у ГГТУ ім. П. О. Сухога ў ручным і аўтаматычным рэжымах

ABSTRACT

The theme The software complex of timetable management in GSTU named after P. O. Sukhoi
The object of study is the timetable at GSTU named after P. O. Sukhoi
The purpose of the work is the development of a software package for managing the schedule at the GSTU named after P. O. Sukhoi which allows you to generate a schedule in automatic mode
The main result of the work is a program that allows you to make schedule at the GSTU named after P. O. Sukhoi in manual and automatic modes

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Расписание занятий в высшем учебном заведении служит для сведения в единую взаимосвязанную систему обучающихся, преподавателей, учебных предметов и назначенных для проведения занятий мест – аудиторий. Оптимизация расписания занятий является одним из основных факторов, способных существенно оптимизировать учебный процесс [1].

На сегодняшний день проблемным остается вопрос о составлении учебного расписания, так как это весьма затратный процесс и в плане времени, и в плане соблюдения всех требований министерства образования [1].

Многие вузы до сих пор используют ручной режим составления расписания: предполагается минимум два методиста на факультет для составления расписания очной и заочной форм обучения. Часто бывает так, что из-за человеческого фактора появляются нестыковки и накладки в аудиторном фонде и между общеуниверситетскими преподавателями. Как правило, такое расписание составляется на листах со сводной сеткой по дням недели, каждая дисциплина и преподаватель вписываются мелким почерком, а далее распространяется по всем кафедрам и факультетам, где каждый преподаватель должен делать для себя выписку с расписанием проведения пар. Усложняется ситуация тем, что многие университеты имеют несколько учебных корпусов, поэтому необходимо учитывать время перемещения студентов и преподавателей между ними. Такие сложности наравне с трудозатратами являются предпосылками к автоматизации ввода и хранения данных. Предлагаемые для учебных заведений программные продукты позволяют оптимально формировать расписание в автоматическом, а не ручном режиме, буквально несколькими операциями [1].

В основе составления расписания занятий лежит теория расписания. Теория расписания является хорошо изученной и описанной во многих работах, начиная с 1960-х годов. Она широко используется как при организации работы предприятий, так и применима для учебных заведений [2][3][4][5].

Целью написания данной работы является разработка программного комплекса управления расписанием в ГГТУ им. П. О. Сухого, позволяющего формировать расписание в автоматическом режиме. А вот какие задачи являются обязательными для достижения цели диплома:

– выполнить обзор и анализ информации по предметной области и по разработке автоматизированных систем управления расписанием, выполнить анализ существующих технологий и систем программирования, систем управления базами данных;

– разработать функциональную модель системы управления расписанием;

– разработать структуру и состав информационного обеспечения для сопровождения программного комплекса

1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ

1.1 Анализ предметной области

1.1.1 Расписание занятий с точки зрения формализации в теории расписаний есть определение на шкале времени места проведения занятий по заданным дисциплинам обучения с выполнением предъявляемых к ним требованиям. Требования формируются участниками учебного процесса и руководящими документами [2].

Исходными составляющими данного процесса являются:

– **P** – потоки обучения, которые включают в свой состав от одной до нескольких групп обучения, или подгруппы, которые образуются за счет деления одной или нескольких групп (поток) на отдельные подразделения;

– **T** – преподаватели, являющиеся основным механизмом воздействия на потоки обучения. В отличие от классических подходов в теории расписания (один механизм - одна операция) в данной ситуации могут быть ситуации, когда несколько преподавателей могут объединяться в один «механизм» для проведения занятия;

– **D** – дисциплины обучения, основой которой является тематический план обучения, включающий различные типы занятий;

– **A** – аудитории, являющиеся местом для проведения занятий (выполнения операции). Во многих существующих теориях и системах, реализующих составление расписания, данный участник выносится из общей задачи с целью упрощения системы.

1.1.2 Постановка задачи на планирование.

Во многих работах она формулируется, как перебор всевозможных вариантов для всех исходных данных процесса:

$$R = \{P * T * D * A * z\}, \quad (1.1)$$

где z – периоды проведения занятия (дата и пара).

При таком подходе делается вывод о сложности составления расписания, так как при ее решении появляется экспоненциальный рост количества сочетаний, что делает задачу NP-полной [2][3].

Однако такой подход является не всегда корректным, так как уже на предварительной подготовке к планированию данное сочетание сокращается за счет объединения преподавателя, потока, аудитории (или возможной аудитории) и проводимого занятия по тематическому плану в одну единицу планирования – называемого часто занятие.

Если рассматривать планирование в данном случае как процесс определения временного отрезка для конкретного занятия, то задача становится классической с точки зрения теории расписаний. То есть, для заданного числа

работ (дисциплин) и операций (занятие дисциплины) определить такие временные отрезки, чтобы составленное расписание соответствовало заданным критериям оптимальности и предъявляемым требованиям.

При этом для составления расписания должны быть определены:

– дисциплины и занятия по ним (работы и операции) – основой является тематический план изучения дисциплины;

– преподаватели и аудитории (машины и место расположения машин) для проведения занятий. В большинстве случаев по всем занятиям в тематическом плане определены преподаватели (жесткая привязка) и возможные аудитории (плавающая привязка);

– порядок прохождения занятий (операций по машинам). В настоящее время данный пункт при составлении расписания занятий опускается на основе допустимого предположения, что при усвоении материалов дисциплины он не является критичным. То есть с точки зрения составления расписания порядок проведения занятий будет являться случайным;

– критерий оценки расписания – некоторый параметр, вычисляемый по полученному расписанию, показывающий его оптимальность с заданных точек зрения. Для учебного процесса критерием оценки расписания выступает многопараметрическая функция, включающая как дискретные обязательные требования, так и рекомендуемые оптимизационные требования. Во многих работах данная функция является определяющей с точки зрения составления оптимального расписания. В качестве ее часто используют функцию суммы штрафов, позволяющей достаточно просто оценить оптимальность составленного расписания.

Для автоматизированного составления расписания достаточно задать первые два пункта. Важным моментом для автоматизации является создание такого подхода, который бы позволял оператору делать расписание по выбранным занятиям за минимальное время с контролем на уровне программы или оператором визуально за выполнением заданных требований. Фактически в данном случае решение задачи сводится к поиску наилучшего интерфейса работы программы и является чисто инженерной задачей. Для облегчения работы оператора на первом шаге автоматизации можно использовать алгоритм динамического программирования. То есть определяется порядок составления расписания по дисциплинам таким образом, чтобы разделить процесс планирования на подзадачи, составление расписания по которым будет являться относительно несложным. При этом основной задачей в этом случае будет являться определение критерия ранжирования дисциплин.

Для данного подхода можно использовать следующий критерий [2][8]:

$$K_{opt}(i,j) = W_{rd}(i)/W_{mo}(j), \qquad (1.2)$$

где i – номер преподавателя (машины), лежащее в диапазоне от 1 до M;

j – номер дисциплины (работы), лежащее в диапазоне от 1 до N;

W_{rd} – количество занятий (операций), которое необходимо провести по выбранной дисциплине (работе);

W_{mo} – количество занятий (операций), которые может провести преподаватель по планируемому периоду времени с учетом наложенных ограничений на выполнение занятий по данной дисциплине и преподавателю.

Ранжирование дисциплин для составления расписания выполняется в порядке уменьшения полученного критерия.

В теории расписаний данный критерий часто обозначают как резерв времени на выполнение работ – разница между количеством времени машины и количеством времени на работу [5]. В данном случае предлагается использовать отношение этих переменных, позволяющее не только расставить дисциплины по порядку планирования, но и проверить также возможность планирования работ по формуле:

$$K_j = Sum\big(K_{opt}(i,j); N\big) < 1, \quad (1.3)$$

где N – количество дисциплин по данному преподавателю.

Если полученный показатель будет больше 1, то это означает, что преподаватель не располагает достаточным временем для проведения занятий. В этом случае необходимо снизить ограничения на проведение данных занятий, либо выполнить замену преподавателя.

Составленное расписание занятий по данному алгоритму будет являться лишь частично оптимальным. Но в большинстве случаев современных подходов к составлению расписания занятий полученное расписание есть некий компромисс при проведении занятий, полученный на основе опыта операторов и требований, предъявляемых к проведению занятий.

При реализации данного подхода необходимо использовать итерационный метод. То есть при составлении расписания после каждой дисциплины выполняется повторное ранжирование работ.

При возникновении ситуации, когда по выбранной очередной дисциплине планирование является невозможным, выполняется перепланирование. В качестве перепланируемой дисциплины выбирается та, у которой наименьшее значение показателя $K_{opt}(i,j)$.

Для решения проблемы ограничений, накладываемых со стороны составляющих процесса планирования возможно использовать рекуррентный алгоритм адаптации процесса планирования.

1.2 Аналитический обзор существующих аналогов

1.2.1 Система «АВТОРасписание» предназначена для быстрого, удобного и качественного составления расписаний занятий и сопровождения их в течение всего учебного года.

Имеется восемь основных модификаций программы для различных учебных заведений: для средних общеобразовательных школ, лицеев и

гимназий; колледжей, техникумов и профессиональных училищ; училищ искусства и культуры; вузов (очная форма обучения); военных вузов; учебных центров, УПК и ИПК; вузов с несколькими удаленными учебными корпусами с учетом времени переезда между ними (очная и заочная формы обучения, сетевая версия).

Интерфейс программы представлен на рисунках 1.1, 1.2.

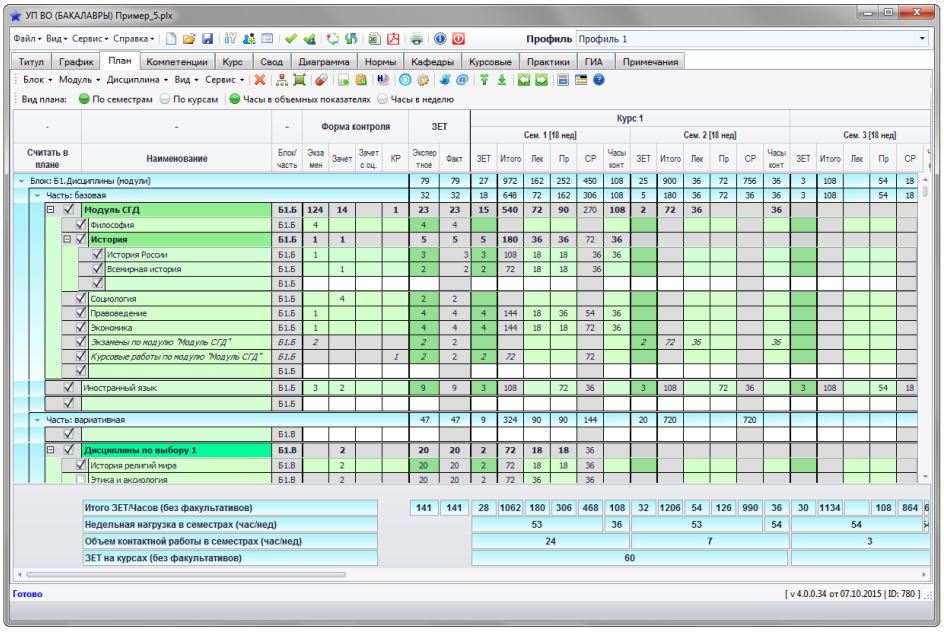

Рисунок 1.1 – Интерфейс программы AVTOR (активная вкладка «План»)

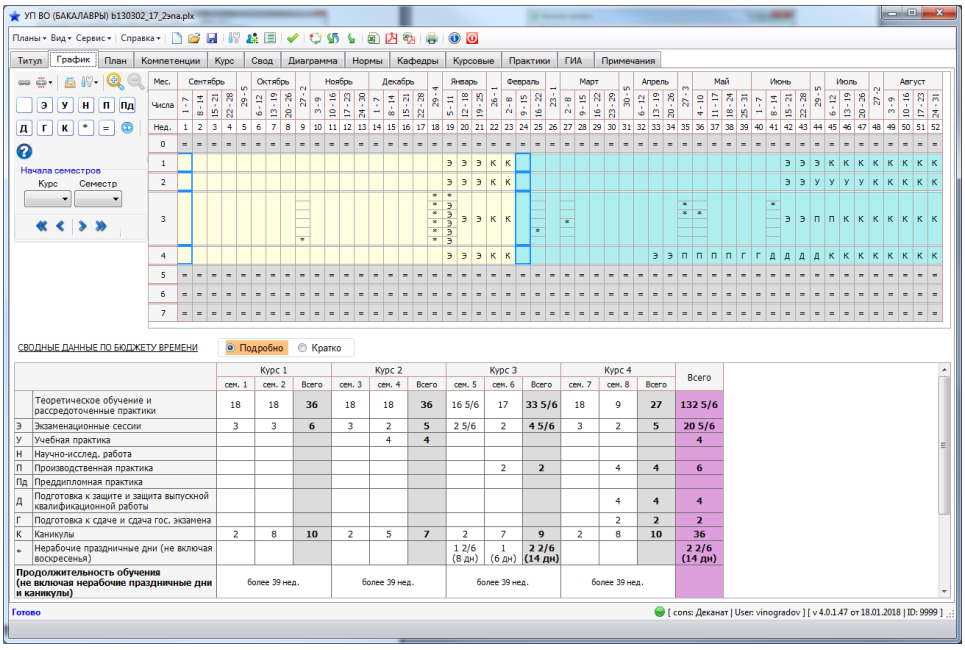

Рисунок 1.2 – Интерфейс программы AVTOR (активная вкладка «График»)

AVTOR позволяет максимально облегчить и автоматизировать сложный труд составителей расписания. Система помогает легко строить, корректировать и распечатывать в виде удобных и наглядных документов:

– расписания занятий классов (учебных групп);

– расписания преподавателей;

– расписания занятости аудиторий (кабинетов);

– учебные нагрузки.

AVTOR-2 имеет приятный дизайн и дружественный сервис. Программа достаточно проста в освоении. Имеется подробное «Руководство пользователя» и справочная система (HELP), где описаны все возможности и способы работы с программой.

Программа отличается уникальным и очень мощным алгоритмом построения и оптимизации расписания. Этот алгоритм является оригинальной авторской разработкой. Он позволяет находить оптимальные решения даже при очень сложных исходных данных.

Полученное автоматическое расписание практически не требует ручной доработки; даже при очень сложных и жестких ограничениях автоматически размещаются все возможные занятия. Если в исходных данных имеются неразрешимые противоречия, то их можно обнаружить и устранить, используя блок анализа.

Технические характеристики: Время работы программы зависит от размерности учебного заведения и мощности компьютера. Полный расчет и оптимизация расписания школы среднего размера со сложными исходными данными (40 классов, 80 преподавателей, из них более 10 совместителей; две смены; дефицит аудиторий) идет около 2-3 минут на компьютере типа Celeron-2000.

AVTOR позволяет:

– строить расписание без «окон» у классов (учебных групп);

– оптимизировать в расписании «окна» преподавателей;

– учитывать требуемый диапазон дней/часов для классов, для преподавателей и для аудиторий;

– учитывать характер работы и пожелания как штатных сотрудников, так и совместителей-почасовиков;

– оптимально размещать занятия по кабинетам (аудиториям) с учетом особенностей классов, предметов, приоритетов преподавателей и вместимости кабинетов;

– вводить расписание звонков;

– устанавливать время перехода (переезда) между учебными корпусами;

– оптимизировать количество переходов из кабинета в кабинет, и из корпуса в корпус;

– легко соединять любые классы (учебные группы) в потоки при проведении любых занятий;

– разделять классы (учебные группы) при проведении занятий по иностранному языку, физической культуре, труду, информатике (и любым другим предметам) на любое количество подгрупп (до десяти!);

– вводить комбинированные уроки для подгрупп (типа «иностранный/информатика») по любым предметам;

– вводить (помимо основных предметов) спецкурсы и факультативы;

– оптимизировать равномерность и трудоемкость расписания;

– легко и быстро вводить и корректировать исходные данные;

– иметь любое количество вариантов расписаний;

– автоматически преобразовывать расписания при изменении базы данных;

– легко сохранять в архивах, копировать и пересылать по E-mail полные базы данных и варианты расписаний;

– быстро вносить любые необходимые корректировки в расписание;

– находить замены временно отсутствующих преподавателей;

– автоматически контролировать расписание, исключая любые «накладки» и противоречия;

– выводить расписания в виде удобных и наглядных документов: текстовых, Word, HTML, а также файлов dBase и книг Excel;

– выставлять готовые расписания в локальной сети и на Интернет-страницах для общего доступа [9].

1.2.2 Система «Электронное расписание» специально разработана с учетом требований учреждений высшего профессионального образования, содержит функции планирования и составления расписания.

Преимущества использования системы:

– мгновенный выбор расписания с любого устройства как для группы, так и для преподавателя. Мгновенное отображение любых изменений;

– при составлении расписания диспетчер сразу видит свободных преподавателей и доступные аудитории, что не допускает случайного наложения занятий;

– успешное внедрение информационных технологий в учебный процесс является важным фактором для повышения имиджа Вашего учебного учреждения [10].

Интерфейс программы «Электронное расписание» отражен на рисунке 1.3.

Рисунок 1.3 – Интерфейс программы «Электронное расписание»

1.2.3 Система «Расписание ПРО» позволяет преподавателям и учителям значительно сократить время составления школьных или институтских расписаний при минимальных затратах усилий.

Основная технология работы представлена в древовидной структуре «Управление» – необходимо пройти последовательно по «веткам» этого дерева сверху вниз. Каждый вариант расписания рассматривается как отдельный проект. Введя однажды исходные данные, можно в дальнейшем делать копию проекта и работать (редактировать и экспериментировать) с ней.

Программа поддерживает два режима управления данными: ручной и автоматический.

Вывести на печать готовое расписание по группам или по преподавателям можно непосредственно из программы, воспользовавшись меню «Файл». Если необходимо дополнительное оформление внешнего вида расписания, можно произвести экспорт результатов в Microsoft Excel.

1.3 Обзор технологий для реализации программного комплекса

Для того, чтобы выбрать стек технологий для реализации программного комплекса, необходимо решить, на каких платформах он будет использоваться. Итоговая программа разрабатывается под ОС *Windows* и является *desktop*-приложением.

В качестве языка программирования выбран *Python3.9*.

Python – это язык программирования с открытым исходным кодом, созданный голландским программистом Гвидо ван Россумом и названный в честь британской труппы комиков «Монти Пайтон» (Monty Python). Одним из ключевых соображений ван Россума было то, что программисты тратят больше времени на чтение кода, чем на его написание, поэтому он решил создать легко

читаемый язык. *Python* является одним из самых популярных и простых в освоении языков программирования в мире [11].

Для того, чтобы ускорить разработку, сократить число ошибок и время, затраченное на их исправление, необходимо выбрать удобную и многофункциональную интегрированную среду разработки (IDE). В качестве IDE выбран *PyCharm*. *PyCharm* делает разработку максимально продуктивной благодаря функциям автодополнения и анализа кода, мгновенной подсветке ошибок и быстрым исправлениям. Автоматические рефакторинги помогают эффективно редактировать код, а удобная навигация позволяет мгновенно перемещаться по проекту [12].

Кроме того, *PyCharm* позволяет установить полезные дополнения. Дополнения обеспечивают:

– подсказки с использование искусственного интеллекта;

– запуск переводчика прямо из кода программы;

– проверка типов переменных в Python (чтобы не допускать ошибки, связанные с типом возвращаемых значений);

– проверка кода программы на уязвимости в автоматическом режиме;

– просмотр файла не открывая его;

– запуск блока кода в консоли без запуска самой программы;

– выделение названий разными цветами (к примеру, чтобы не забыть отредактировать выделенный фрагмент);

– изменение вида редактора в целом;

– и многое другое.

В качестве системы управления базами данных выбрана *PostgreSQL*.

PostgreSQL является одной из самых популярных баз данных. За более чем 20-летнюю историю развития на прочном фундаменте, заложенном академической разработкой, *PostgreSQL* выросла в полноценную СУБД уровня предприятия и составляет реальную альтернативу коммерческим базам данных.

Вопросы обеспечения надежности особенно важны в приложениях уровня предприятия для работы с критически важными данными. С этой целью *PostgreSQL* позволяет настраивать горячее резервирование, восстановление на заданный момент времени в прошлом, различные виды репликации (синхронную, асинхронную, каскадную).

PostgreSQL позволяет работать по защищенному SSL-соединению и предоставляет большое количество методов аутентификации, включая аутентификацию по паролю, клиентским сертификатам, а также с помощью внешних сервисов (LDAP, RADIUS, PAM, Kerberos).

При управлении пользователями и доступом к объектам БД предоставляются следующие возможности:

– создание и управление пользователями и групповыми ролями;

– разграничение доступа к объектам БД на уровне как отдельных пользователей, так и групп;

– детальное управление доступом на уровне отдельных столбцов и строк;

– поддержка *SELinux* через встроенную функциональность *SE-PostgreSQL* (мандатное управление доступом).

PostgreSQL обеспечивает полную поддержку свойств ACID и обеспечивает эффективную изоляцию транзакций. Для этого в *PostgreSQL* используется механизм многоверсионного управления одновременным доступом (MVCC). Он позволяет обходиться без блокировок во всех случаях, кроме одновременного изменения одной и той же строки данных в нескольких процессах. При этом читающие транзакции никогда не блокируют пишущие транзакции, а пишущие – читающих. Это справедливо и для самого строгого уровня изоляции *serializable*, который, используя инновационную систему *Serializable Snapshot Isolation*, обеспечивает полное отсутствие аномалий серализации и гарантирует, что при одновременном выполнении транзакций результат будет таким же, как и при последовательном выполнении [13].

Для написания графического интерфейса выбран фреймворк *Kivy*.

Kivy был создан в 2011 году. Данный кросс-платформенный фреймворк *Python* работает на *Windows, Mac, Linux, Android, IoS* и *Raspberry Pi*. В дополнение к стандартному вводу через клавиатуру и мышь он поддерживает мультитач. *Kivy* даже поддерживает ускорение GPU своей графики, что во многом является следствием использования *OpenGL ES2*. У проекта есть лицензия MIT, поэтому библиотеку можно использовать бесплатно и вкупе с коммерческим программным обеспечением.

Во время разработки приложения через *Kivy* создается интуитивно понятный интерфейс (Natural user Interface), или NUI. Его главная идея в том, чтобы пользователь мог легко и быстро приспособиться к программному обеспечению без чтения инструкций [14].

Кроме того, *Kivy* позволяет создавать приложения с отзывчивым дизайном (расположение и наличие элементов управления зависит от размера экрана).

Сообщество *Kivy* разработало множество библиотек, таких как *HotReloader, KivyMD, Awesome KivyMD, AsyncKivy* и др. Перечисленные библиотеки позволяют ускорить разработку и сделать внешний вид приложения современным.

При разработке программного продукта очень важно протестировать его функциональность. В качестве фреймворка для тестирования выбран *pytest*.

Фреймворк *pytest* помогает легко писать небольшие тесты и масштабируется для поддержки сложного функционального тестирования приложений и библиотек.

Возможности *pytest*:

– подробный разбор упавших проверок assert;

– автообнаружение тестовых модулей и функций;

– использование модульных фикстур для управления небольшими или параметризованными тестовыми ресурсами;

– запуск тестовых наборов, написанных с использованием *unittest* (включая пробные) и *nose*;

– совместим с Python 3.5+ и PyPy 3;

– у *pytest* есть большой набор (более 315 внешних плагинов) и процветающее сообщество.

Сегодня большое число программ доставляются и запускаются в так называемых контейнерах.

Контейнеризация (виртуализация на уровне ОС) – технология, которая позволяет запускать программное обеспечение в изолированных на уровне операционной системы пространствах. Контейнеры являются наиболее распространенной формой виртуализации на уровне ОС. С помощью контейнеров можно запустить несколько приложений на одном сервере (хостовой машине), изолируя их друг от друга.

Процесс, запущенный в контейнере, выполняется внутри операционной системы хостовой машины, но при этом он изолирован от остальных процессов. Для самого процесса это выглядит так, будто он единственный работает в системе [15]

Схема устройства контейнеров представлена на рисунке 1.4.

Рисунок 1.4 – Схема устройства контейнеров

Docker – это программная платформа для быстрой разработки, тестирования и развертывания приложений. *Docker* упаковывает ПО в стандартные блоки, которые называются контейнерами.

В рамках диплома технология *Docker* будет использоваться для запуска СУБД *PostgreSQL* и хранения данных, а также для запуска тестов.

При написании программного кода рекомендуется придерживаться определенного формата. В *Python* существует *PEP 8* – руководство по написанию кода. Для того, чтобы форматировать код в соответствии со стандартом, будет использоваться технология *Black.*

Black – это фреймворк, который может автоматически изменять форматирование *python*-файлов.

В качестве системы контроля версий выбрана *Git*.

Система контроля версий (СКВ) – это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определенным старым версиям этих файлов.

1.4 Постановка задачи на дипломное проектирование

Целью написания данной работы является разработка программного комплекса управления расписанием в ГГТУ им. П. О. Сухого, позволяющего формировать расписание в автоматическом режиме.

В результате проведенного аналитического обзора существующих аналогов были сформулированы следующие функциональные требования к проектируемому программному обеспечению:

– строить расписание без «окон» у учебных групп;

– оптимизировать в расписании «окна» преподавателей;

– учитывать требуемый диапазон дней/часов для классов, для преподавателей и для аудиторий;

– учитывать характер работы и пожелания преподавателей;

– оптимально размещать занятия по кабинетам (аудиториям) с учетом особенностей учебных групп, предметов, приоритетов преподавателей и вместимости кабинетов;

– вводить расписание звонков;

– в ручном режиме заполнения расписания иметь возможность просмотра расписания одновременно в двух видах (всего четыре вида: расписание преподавателей, заполнение аудиторий, расписание учебных групп, окно рабочих нагрузок);

– устанавливать время перехода (переезда) между учебными корпусами;

– оптимизировать количество переходов из кабинета в кабинет, и из корпуса в корпус;

– легко соединять любые учебные группы в потоки при проведении любых занятий;

– разделять учебные группы при проведении занятий по иностранному языку, физической культуре (и любым другим предметам) на любое количество подгрупп;

– вводить комбинированные уроки для подгрупп (типа «иностранный/информатика») по любым предметам;

– легко и быстро вводить и корректировать исходные данные;

– автоматически преобразовывать расписания при изменении базы данных;

– быстро вносить любые необходимые корректировки в расписание;

– находить замены временно отсутствующих преподавателей;

– автоматически контролировать расписание, исключая любые «накладки» и противоречия;

– выводить расписания в виде PDF-документов, готовых к экспорту на сайт университета.

Исходными данными к проекту являются:

– база данных университета (нужны не все таблицы, а только те, которые связаны с расписанием);

– наряды на преподавателей;

– ФИО ответственных лиц за составление и внедрение расписания;

– расписание звонков.

Предполагаемыми выходными данными являются:

– PDF-документы с расписанием.

В силу того, что приложение оперирует открытыми данными (которые можно найти на официальном сайте университета ГГТУ им. П. О. Сухого), оно может быть установлено на любую машину. С точки зрения безопасности, важными данными являются имя пользователя и пароль в PostgreSQL. Они могут быть получены лишь при непосредственном доступе к машине, на которой установлена программа.

Системные требования для работы приложения: Windows 7/10, 4 Гб ОЗУ, 1000 Мб на жестком диске, разрешение экрана 1920 × 1080.

Разрабатываемое приложение должно вести журнал действий пользователя и обрабатывать любые пользовательские ошибки. Выводить соответствующие совершенным действиям уведомления и, если требуется, подсказки. Допускается длительная работа (более одного астрономического часа) программы при составлении расписания в силу сложности процесса и отсутствии универсального алгоритма.

2 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Функциональное моделирование

2.1.1 В настоящее время для разработки функциональных моделей программного обеспечения используется унифицированный язык моделирования.

Унифицированный язык моделирования (UML) – это семейство графических нотаций, в основе которого лежит единственная метамодель. Он помогает в описании и проектировании программных систем, в особенности систем, построенных с использованием объектно-ориентированных (ОО) технологий [6].

UML – точный язык, поэтому можно было бы ожидать, что он основан на предписывающих правилах. Но UML часто рассматривают как программный эквивалент чертежей из других инженерных дисциплин, а эти чертежи основаны не на предписывающих нотациях. Никакой комитет не говорит, какие символы являются законными в строительной технической документации; эти нотации были приняты по соглашению, как и в естественном языке [6].

Основным компонентом языка UML является диаграмма.

Диаграмма (diagram) – графическое представление совокупности элементов модели в форме связного графа, вершинам и ребрам (дугам) которого приписывается определенная семантика.

В нотации языка UML определены следующие виды канонических диаграмм:

– вариантов использования (use case diagram);
– классов (class diagram);
– кооперации (collaboration diagram);
– последовательности (sequence diagram);
– состояний (statechart diagram);
– деятельности (activity diagram);
– компонентов (component diagram);
– развертывания (deployment diagram).

Диаграммы представляют статическую структуру приложения (диаграммы вариантов использования, классов и др.), поведенческие аспекты разрабатываемой программной системы (диаграмма деятельности), физические аспекты функционирования системы (диаграммы реализации).

Еще одним компонентом языка является сущность. К сущностям UML относятся структурные сущности (классы, интерфейсы, прецеденты), поведенческие сущности (автоматы), групповые сущности (пакеты), аннатационные сущности (примечания) [7].

2.1.2 Разрабатываемый программный комплекс предназначен для управления расписанием университета и предоставляет диспетчерам возможность перехода от ручного режима составления расписания к автоматическому.

В составлении расписания выделяют следующие роли: диспетчер, методист кафедры, методист деканата, преподаватель.

Подразумевается, что база данных университета содержит все необходимые сведения о преподавателях, группах, дисциплинах и аудиториях в университете, т.к. именно база данных является источником исходных данных для автоматического формирования расписания.

Для того, чтобы внедрить программный комплекс, необходимо, чтобы методист кафедры имел частичный доступ к базе данных университета и имел права на добавление/удаление/редактирование записей таблицы, хранящей сведения о нарядах на преподавателей. Пожелания преподавателей также добавляются в виде записей в эту таблицу и будут учтены в дальнейшем при составлении расписания.

Функциональная модель «Составление расписания» может быть описана при помощи диаграммы вариантов использования, которая представлена на рисунке 2.1.

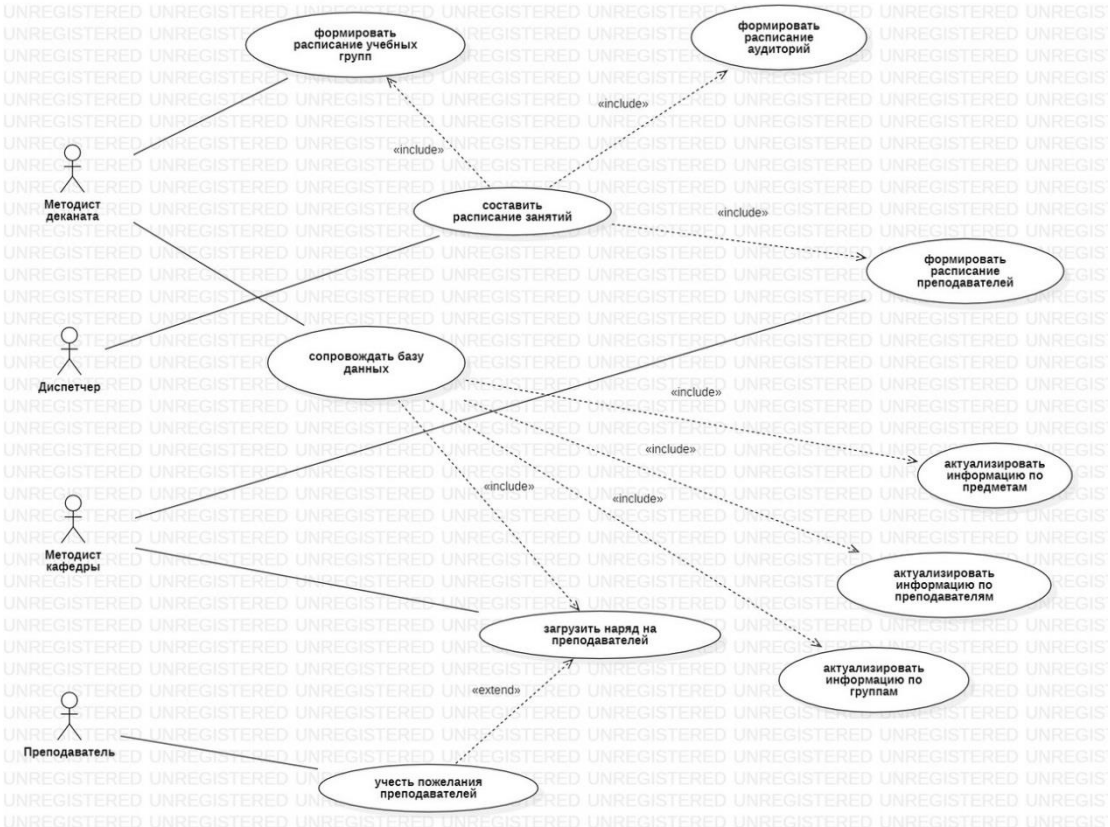

Рисунок 2.1 – Диаграмма вариантов использования функциональной модели «Составление расписания»

На рисунке 2.2 представлена диаграмма IDEF0 функциональной модели «Составление расписания».

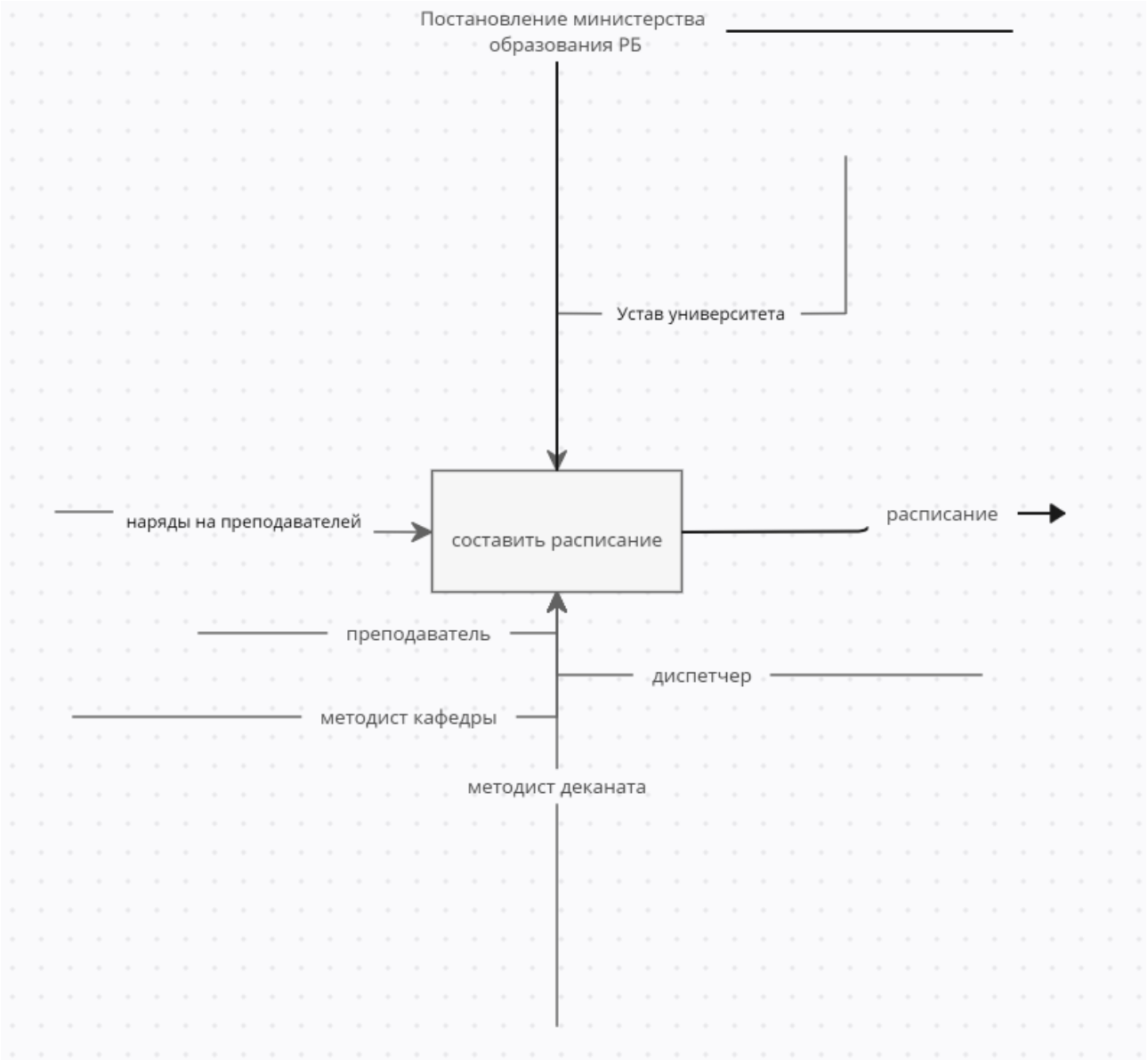

Рисунок 2.2 – Диаграмма IDEF0 функциональной модели «Составление расписания»

2.2 Разработка информационной модели

В результате анализа предметной области, а также ее функциональной модели, были выделены следующие классы, представляющие систему в целом:

– факультет;
– кафедра;
– учебная группа;
– преподаватель;

– предмет;
– тип предмета;
– расписание;
– аудитория;
– рабочая нагрузка;
– запись расписания;
– аудитория занятия.

Класс «факультет» описывает факультеты университета и содержит следующие поля:

– наименование;
– глава.

Класс «кафедра» описывает кафедры университета и содержит следующие поля:

– наименование;
– код факультета;
– глава.

Класс «учебная группа» описывает учебные группы и содержит следующие поля:

– наименование;
– число студентов;
– код факультета.

Класс «преподаватель» описывает преподавателей и содержит следующие поля:

– ФИО;
– ученая степень;
– код кафедры.

Класс «предмет» описывает учебные дисциплины и содержит следующие поля:

– наименование.

Класс «тип предмета» описывает типы учебных дисциплин и содержит следующие поля:

– наименование.

Класс «расписание» описывает расписания и содержит следующие поля:

– год;
– семестр.

Класс «аудитория» описывает аудитории и содержит следующие поля:

– номер;
– число посадочных мест;
– код кафедры.

Класс «рабочая нагрузка» описывает аудитории и содержит следующие поля:

– код группы;
– код преподавателя;
– код предмета;
– код типа предмета;
– год;
– семестр;
– количество часов в неделю.

Класс «запись расписания» описывает аудитории и содержит следующие поля:

– код группы;
– код преподавателя;
– код предмета;
– код типа предмета;
– код аудитории;
– код расписания;
– номер пары;
– день недели;
– тип недели;
– подгруппа;
– свободен преподаватель.

2.3 Проектирование базы данных

Для проектирования базы данных существуют различные методики, различные последовательности шагов или этапов. В целом, можно выделить следующие этапы:

– выделение сущностей и их атрибутов, которые будут храниться в базе данных, и формирование по ним таблиц;
– определение уникальных идентификаторов (первичных ключей) объектов, которые хранятся в строках таблицы;
– определение отношений между таблицами с помощью внешних ключей;
– нормализация базы данных.

В настоящее время для упрощения работы с базами данных используются ORM-библиотеки. Они позволяют взаимодействовать с базами данных с помощью объектно-ориентированного кода, не используя SQL-запросы.

Одной из самых популярных ORM-библиотек для Python сегодня является *SQLAlchemy*. У нее есть особенность – код приложения будет оставаться тем же

вне зависимости от используемой базы данных. Это позволяет с легкостью мигрировать с одной базы данных на другую, не переписывая код приложения.

С выходом последней версии *SQLAlchemy* (версия 2.0) появилась возможность создания универсальных классов, позволяющих одновременно описывать сущности, их атрибуты, связи и ограничения, а также использовать экземпляры данных классов в качестве объектов-значений.

На рисунке 2.3 приведена логическая модель спроектированной базы данных для программного комплекса.

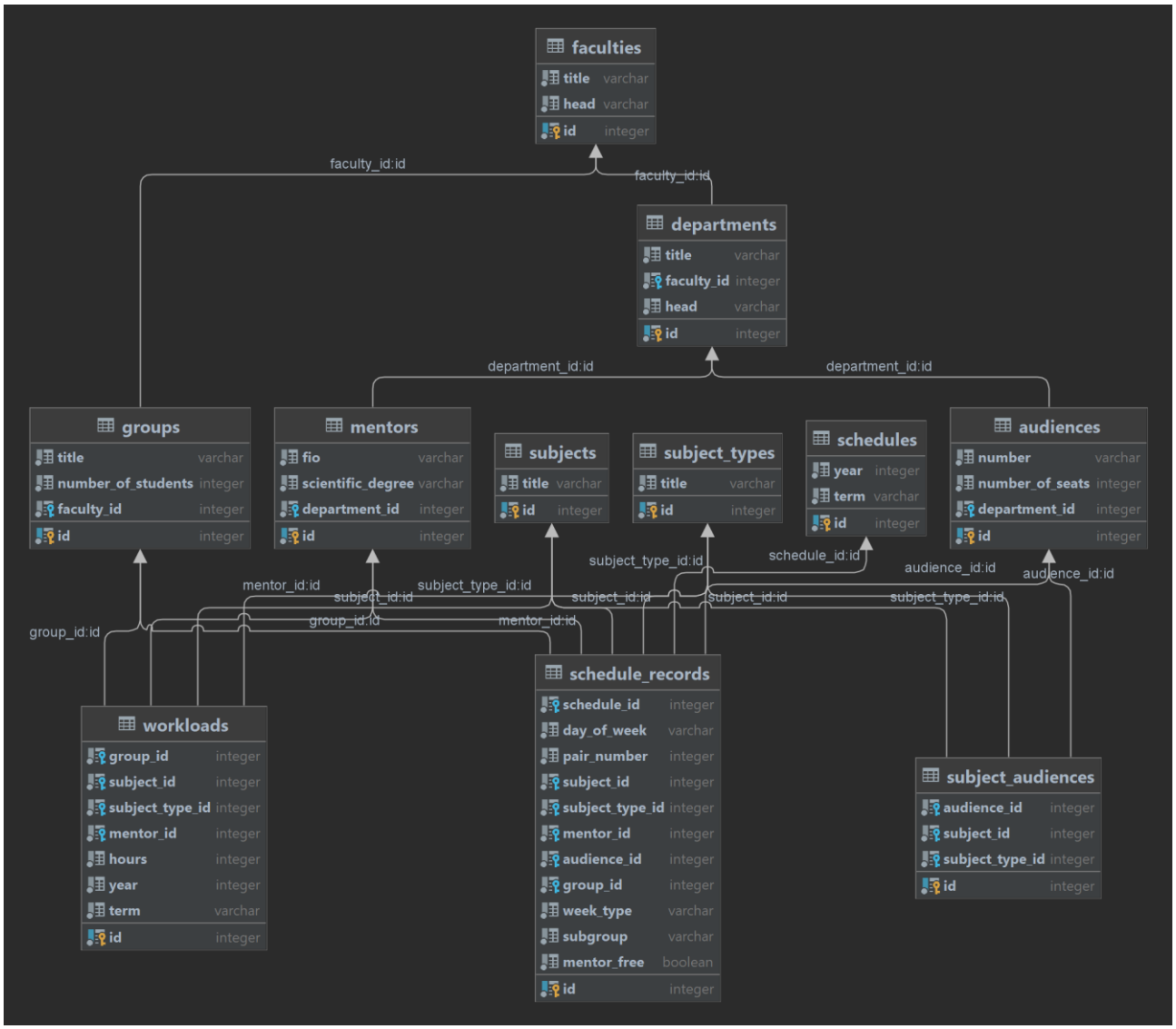

Рисунок 2.3 – Логическая модель спроектированной базы данных для программного комплекса

Сущность «schedules» представляет собой информацию о имеющихся расписаниях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.1.

Таблица 2.1 – Таблица «schedules»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
year	varchar	Да	Год
term	varchar	Да	Семестр

Сущность «audiences» представляет собой информацию о имеющихся аудиториях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.2.

Таблица 2.2 – Таблица «audiences»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
number	varchar	Да	Номер аудитории
number_of_seats	integer	Да	Число посадочных мест
department_id	integer	Нет	Внешний ключ (кафедра)

Сущность «departments» представляет собой информацию о имеющихся кафедрах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.3.

Таблица 2.3 – Таблица «departments»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
title	varchar	Да	Название
faculty_id	integer	Нет	Внешний ключ (факультет)
head	varchar	Да	ФИО зав. кафедрой

Сущность «faculty» представляет собой информацию о имеющихся факультетах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.4.

Таблица 2.4 – Таблица «faculty»

Имя столбца	Тип данных	Разрешение NULL	Описание
id	integer	Нет	Первичный ключ
title	varchar	Да	Название
head	integer	Да	ФИО декана

Сущность «groups» представляет собой информацию о имеющихся учебных группах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.5.

Таблица 2.5 – Таблица «groups»

Имя столбца	Тип данных	Разрешение NULL	Описание
id	integer	Нет	Первичный ключ
title	varchar	Да	Название
number_of_students	integer	Да	Число студентов
faculty_id	integer	Нет	Внешний ключ (факультет)

Сущность «mentors» представляет собой информацию о имеющихся преподавателях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.6.

Таблица 2.6 – Таблица «mentors»

Имя столбца	Тип данных	Разрешение NULL	Описание
id	integer	Нет	Первичный ключ
fio	varchar	Да	ФИО
scientific_degree	varchar	Да	Научная степень
department_id	integer	Нет	Внешний ключ (кафедра)

Сущность «schedule_records» представляет собой информацию о имеющихся записях расписания. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.7.

Таблица 2.7 – Таблица «schedule_records»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
schedule_id	integer	Нет	Внешний ключ (расписание)
day_of_week	varchar	Нет	День недели
pair_number	integer	Да	Номер пары
subject_id	integer	Нет	Внешний ключ (предмет)
subject_type_id	integer	Нет	Внешний ключ (тип предмета)
mentor_id	integer	Нет	Внешний ключ (преподаватель)
audience_id	integer	Нет	Внешний ключ (аудитория)
group_id	integer	Нет	Внешний ключ (группа)
week_type	varchar	Да	Тип недели
subgroup	varchar	Да	Подгруппа
mentor_free	bool	Да	Если False, то означает невозможность проведения занятия преподавателем

Сущность «subject» представляет собой информацию о имеющихся предметах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.8.

Таблица 2.8 – Таблица «subject»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	Integer	Нет	Первичный ключ
title	varchar	Да	Название

Сущность «subject_audiences» представляет собой информацию о возможности проведения занятий в определенных аудиториях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.9.

Таблица 2.9 – Таблица «subject_audiences»

Имя столбца	Тип данных	Разрешение NULL	Описание
id	integer	Нет	Первичный ключ
subject_id	integer	Нет	Внешний ключ (предмет)
subject_type_id	integer	Нет	Внешний ключ (тип предмета)
audience_id	integer	Нет	Внешний ключ (аудитория)

Сущность «subject_types» представляет собой информацию о имеющихся предметах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.10.

Таблица 2.10 – Таблица «subject_types»

Имя столбца	Тип данных	Разрешение NULL	Описание
id	Integer	Нет	Первичный ключ
title	varchar	Да	Название

Сущность «workloads» представляет собой информацию о имеющихся рабочих нагрузках (берутся из нарядов преподавателей). Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.11.

Таблица 2.11 – Таблица «workloads»

Имя столбца	Тип данных	Разрешение NULL	Описание
id	integer	Нет	Первичный ключ
subject_id	integer	Нет	Внешний ключ (предмет)
subject_type_id	integer	Нет	Внешний ключ (тип предмета)
mentor_id	integer	Нет	Внешний ключ (преподаватель)
group_id	integer	Нет	Внешний ключ (группа)
hours	integer	Да	Количество часов в неделю
year	integer	Да	Год
term	varchar	Да	Семестр

3 СТРУКТУРА И ОСНОВНЫЕ АЛГОРИТМЫ СОЗДАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Паттерны проектирования и структура программы

Паттерны проектирования и выделение структуры приложения на модули помогают сделать код более организованным, переиспользуемым и легким для понимания. Использование паттернов проектирования позволяет разработчикам использовать проверенные, оптимальные способы решения задач, которые были созданы опытными разработчиками. Это помогает сделать код более структурированным и гибким для изменений и поддержки в будущем. Принцип разбиения приложения на модули помогает организовать код в логичную и четкую структуру. Разработчики могут легко находить нужный код и добавлять новые функции, минимизируя риск пересечения и конфликта существующего кода. В целом, использование паттернов проектирования и разбиение приложения на модули позволяет значительно улучшить качество и эффективность разработки, а также облегчить поддержку и дальнейшее развитие приложения.

Пользуясь философией программирования Дзен Пайтона [17], принципами SOLID [18], рекомендациями из книги о паттернах разработки на Python [16], было принято решение создать приложение со слабосвязной архитектурой. Слабосвязность кода имеет ряд преимуществ перед традиционным, сильно связанным кодом:

– код с слабосвязной архитектурой легче изменять и обновлять, поскольку изменения в одной части кода не приводят к сбоям в других частях, что особенно полезно при разработке крупных и сложных систем;

– когда компоненты мало связаны между собой, они могут быть перенесены в другие проекты или системы, что упрощает переносимость кода;

– слабосвязная архитектура облегчает тестирование отдельных компонентов и модулей, что помогает сократить время, затраченное на поиск и устранение ошибок;

– слабосвязный код облегчает параллельное программирование, поскольку компоненты могут работать независимо друг от друга;

– слабосвязный код проще масштабировать, поскольку его компоненты могут быть легко заменены или добавлены, не влияя на остальные компоненты.

Слабосвязность достигается посредством применения инвертирования зависимостей, внедрения абстракций.

В программе использован следующий набор паттернов проектирования:

- *репозиторий*;
- *сервисный слой*;
- *события предметной области*;
- *шина сообщений*;
- *обработчик*.

Каждый из этих паттернов выполняет конкретную функцию в архитектуре приложения.

Паттерн «Репозиторий» представляет упрощающую абстракцию хранения данных, которая позволяет устранить связанность слоя модели и слоя данных [16].

Паттерн «Сервисный слой» осуществляет работу с бизнес-логикой, предоставляя интерфейс для взаимодействия с ней из основного приложения. Этот паттерн позволяет упростить код и разделить бизнес-логику от конкретной реализации приложения.

Паттерн «События предметной области» – это способ передать идею о том, что некоторые взаимодействия с системой являются триггерами для других [16]. Он позволяет взаимодействовать с бизнес-логикой при помощи доменных событий, что упрощает взаимодействие с системой и уменьшает сложность кода.

«Шина сообщений» используется для того, чтобы дать действиям возможность запускать события и вызывать надлежащие «Обработчики» [16]. Используется для обработки сообщений (команд и событий) между разными сервисами и слоями приложения.

Приложение имеет луковичную архитектуру, т.е. модель предметной области находится «внутри» и зависимости (слой визуализации, слой базы данных) «втекают» в нее. Также архитектуру можно описать как событийно-управляемую: взаимодействие с базой данных осуществляется посредством отправления команд в шину сообщений. Возвращаемое из шины сообщений значение является событием.

Команды и события являются классами, предназначенными для хранения данных - датаклассами. Для того, чтобы считаться командой, датакласс должен быть наследником базового класса *Command*. То же самое касается события, только датакласс должен быть наследником базового класса *Event*.

Существует множество команд:

- *CreateLocalScheduleRecord*: создать локальную запись расписания;
- *CreateSchedule*: создать расписание;
- *DeleteLocalScheduleRecords*: удалить локальную запись расписания;
- *DeleteSchedule*: удалить расписание;
- *Get10Schedules*: получить 10 расписаний;
- *GetAudiencesForScheduleItem*: получить аудитории для записи распсиания;
- *GetExtendedScheduleRecords*: получить расширенные записи расписания;
- *GetGroupsForScheduleItem*: получить группы для записи расписания;
- *GetMentorsForScheduleItem*: получить преподавателей для записи расписания;
- *GetRowWorkloads*: получить рабочие нагрузки;
- *GetSchedules*: получить расписания;
- *GetSubjectTypesForScheduleItem*: получить типы занятий для записи расписания;
- *GetSubjectsForScheudleItem*: получить предметы для записи расписания;

– *GetUniqueAudiencesDependingOnDepartment*: получить уникальные аудитории в зависимости от кафедры;
– *GetUniqueDepartments*: получить уникальные кафедры;
– *GetUniqueFaculties*: получить уникальные факультеты;
– *GetUniqueGroups*: получить уникальные группы;
– *GetUniqueGroupsDependingOnFaculty*: получить уникальные группы в зависимости от факультета;
– *GetUniqueMentors*: получить уникальных преподавателей;
– *GetUniqueMentorsDependingOnDepartment*: получить уникальных преподавателей в зависимости от кафедры;
– *GetUniqueSubjectTypes*: получить уникальные типы занятий;
– *GetUniqueSubjects*: получить уникальные занятия;
– *GetUniqueTermsDependingOnSchedule*: получить уникальные семестры в зависимости от расписания;
– *GetUniqueTermsDependingOnWorkload*: получить уникальные семестры в зависимости от рабочей нагрузки;
– *GetUniqueYearsDependingOnSchedule*: получить уникальные годы расписаний в зависимости от расписаний;
– *GetUniqueYearsDependingOnWorkload*: получить уникальные годы расписаний в зависимости от рабочей нагрузки;
– *GetWorkloads*: получить рабочие нагрузки;
– *MakeGlobalScheudleRecordsLikeLocal*: сделать глобальные записи расписания как локальные;
– *MakeLocalScheduleRekordsLikeGlobal*: сделать локальные записи расписания как глобальные.

Список существующих событий в приложении:
– *GotAudiencesEntities*: получены сущности аудиторий;
– *GotExtendedScheduleRecords*: получены расширенные записи расписания;
– *GotGroupsEntities*: получены сущности групп;
– *GotMentorsEntities*: получены сущности преподавателей;
– *GotRowWorkloads*: получены рабочие нагрузки;
– *GotSchedules*: получены расписания;
– *GotSubjectEntities*: получены сущности предметов;
– *GotUniqueAudiences*: получены уникальные аудитории;
– *GotUniqueDepartments*: получены уникальные кафедры;
– *GotUniqueFacultis*: получены уникальные факультеты;
– *GotUniqueGroups*: получены уникальные группы;
– *GotUniqueMentors*: получены уникальные преподаватели;
– *GotUniqueSubjectTypes*: получены уникальные типы предметов;
– *GotUniqueSubjects*: получены уникальные предметы;
– *GotIUniqueTerms*: получены уникальные семестры;
– *GotUniqueYears*: получены уникальные годы расписаний;
– *GotWorkloads*: получены рабочие нагрузки;
– *ScheduleIsCreated*: расписание создано;

– *ScheduleIsDeleted*: расписание удалено.

Помимо команд и событий в приложении активно используются перечисления:

– *DayOfWeek*: день недели;

– *Subgroup*: подгруппа;

– *SubjectColor*: цвет занятия (используется при отображении занятий);

– *SubjectType*: тип занятия;

– *Term*: семестр;

– *ViewState*: состояние ячейки расписания;

– *ViewType*: вид таблицы расписания;

– *WeekType*: тип недели.

В процессе проектирования приложения были выделены расширенные сущности, поля которых необходимы для составления расписания. Выделение подобных сущностей существенно сокращает число запросов к базе данных, что способствует оптимизации программного продукта. Расширенные сущности:

– *AdditionalPart*: содержит список id используемых записей расписания из локальной таблицы в базе данных, а также флаг определяющий, занят ли преподаватель;

– *AudiencePart*: содержит id аудитории, ее номер и число мест для сидения;

– *CellPart*: содержит тип недели и подгруппу;

– *CellPos*: хранит день недели и номер пары;

– *GroupPart*: хранит id группы, ее наименование и число студентов;

– *MentorPart*: содержит id преподавателя, его ФИО и ученую степень;

– *ScheduleItemInfo*: содержит в себе все расширенные сущности (они указаны как *nullable*, т.е. поле может содержать *None*-объект, а расширенных частей групп может быть несколько);

– *SubjectPart*: содержит id и наименование для предмета и типа предмета.

В приложении присутствуют интерфейсы:

– AbstractScheduleWeeksStore: существует несколько типов таблиц расписания, каждый из которых состоит из элементов «Неделя», «День», «Пара», «Ячейка пары» (элементы вложены друг в друга) и является хранилищем элементов «Неделя», а также реализует данный интерфейс, гарантирующий передачу актуальной информации вложенным элементам;

– AbstractSizeMaster: для того, чтобы элементы имели пропорциональные размеры, существует интерфейс для объектов, которые выравнивают подконтрольные им объекты;

– AbstractSizeSlave: интерфейс выравнивания объектов требует интерфейса для выравниваемых объектов;

– AbstractTunedByInfoRecords: схожий интерфейс с AbstractScheduleWeeksStore, используется с той же целью, но к объектам, не являющимся хранилищами «Недель»;

– AbstractRepository: реализует паттерн «Репозиторий»;

– AbstractAutoCompleteElement: для удобства и валидации данных разработаны пользовательские виджеты, предлагающие выбрать опцию из выпадающего списка и реализующие данный интерфейс.

К основным классам относится «ScheduleMaster» – класс, предназначенный для хранения актуальной (с учетом уже расставленных занятий) рабочей нагрузки преподавателей. Изначально инициализируются поля: идентификатор расписания, год расписания и семестр (метод update_metadata). Далее устанавливаются все рабочие нагрузки (метод set_workloads), после чего они преобразуются в актуальные с учетом расставленных расписаний (метод fit_workloads_using_info_records). Помимо указанных выше методов, класс предоставляет следующие методы:

– check_if_gropus_fit_in_the_audience: для проверки, поместятся ли группы в указанную аудиторию;

– check_if_groups_workloads_have_enough_hours: прежде, чем назначать группе занятие, необходимо убедиться в том, что это занятие уместно с точки зрения рабочих нагрузок преподавателей;

– convert_cell_part_to_hours: преобразует класс *CellPart* в учебные часы;

– get_old_hours: при изменении уже заполненной ячейки расписания необходимо игнорировать первоначальную запись данной ячейки, но в связи с тем, что актуальные рабочие нагрузки рассчитаны с учетом данной записи, то ее необходимо “приплюсовывать” к имеющимся актуальным рабочим нагрузкам преподавателей;

– get_extended_actual_mentors: получить список типа MentorPart с учетом текущих рабочих нагрузок преподавателей и преподавателя данной записи;

– get_extended_actual_groups: получить список типа GroupPart с учетом текущих рабочих нагрузок преподавателей и групп данной записи.

Ключевым с точки зрения бизнес-логики приложения является класс «Controller». Все его методы (35 штук) оборачиваются декоратором высшего порядка «use_loop» с параметром «use_loading_modal_view», позволяющим на время работы метода создать диалоговое окно, уведомляющее пользователя о том, что запущен какой-либо метод и ему необходимо дождаться его выполнения. Чаще всего методы данного класса принимают в качестве одного из параметров элемент пользовательского интерфейса, который необходимо обновить (в теле методов вызываются методы виджета). Контроллер обращается к шине сообщений, формируя команды из поступающих в него данных.

3.2 Основные алгоритмы

При составлении расписания важно учитывать ограничения на выбираемые варианты в соответствии с актуальной рабочей нагрузкой преподавателей (с учетом уже выставленных занятий). Процесс заполнения расписания является итерационным и требует написания запросов к базе данных для проверки выполнения обязательных условий, а также обращения к методам класса «ScheduleMaster».

Для добавления новой записи расписания необходимо заполнить следующие поля:

- ФИО преподавателя;
- аудитория занятия;
- наименование занятия;
- тип занятия;
- день недели;
- номер пары;
- тип недели;
- подгруппа;
- группы.

Выбор вариантов для заполнения для большинства из приведенных полей должен зависеть от состояния остальных полей.

Алгоритм для получения списка преподавателей:

1. Проверить, помещаются ли выбранные группы в указанной аудитории (в зависимости от того, какая подгруппа указана, от каждой группы берется либо половина студентов, либо вся группа).
2. Убедиться в том, что у всех указанных групп должно проходить данное занятие и есть достаточно нераспределенных часов на это занятие.
3. В случае невыполнения хотя бы одного из вышеупомянутых условий возвращается пустой список.
4. Получить список преподавателей, которые не проводят других занятий на указанной паре в этот день и свободны.
5. Добавить в список только тех преподавателей, которые есть в актуальных рабочих нагрузках, добавив к ним преподавателя, который изначально был указан в текущей ячейке (если она была заполнена перед редактированием).

Алгоритм для получения списка аудиторий:

1. Получить аудитории, в которых можно проводить данное занятие и которые свободны на указанной паре в этот день (аудитория считается свободной в двух случаях: в ней вообще не проводится занятие; проводится выбранное занятие указанным преподавателем).
2. Для каждой из полученных аудиторий осуществить проверку, поместятся ли в нее все указанные группы (в зависимости от того, какая подгруппа указана, от каждой группы берется либо половина студентов, либо вся группа).
3. Все аудитории, позволяющие разместить студентов всех указанных групп, образуют список доступных аудиторий.

Алгоритм для получения списка групп:

1. Получить список всех групп.
2. Оставить только те группы, для которых существуют рабочие нагрузки преподавателей с указанными преподавателями, занятиями и видами занятий.
3. Оставить только те группы, которые есть в актуальных нагрузках преподавателей.

4. Для оставшихся групп проверить, поместятся ли они в указанной аудитории (в зависимости от того, какая подгруппа указана, от каждой группы берется либо половина студентов, либо вся группа), а также убедиться в том, что у группы достаточно нераспределенных часов данного занятия.

Алгоритм для получения списка предметов:

1. Получить список предметов, для которых существуют рабочие нагрузки с указанными преподавателями (проверяется возможность проведения занятия в указанной аудитории).
2. Убедиться в том, что для всех групп достаточно нераспределенных часов данного занятия.

Алгоритм для автоматического составления расписания:

1. Получить два списка рабочих нагрузок: актуальных и всех (используя *ScheduleMaster*).
2. Разделить оба списка по группам в зависимости от учебной группы.
3. Для каждой группы выполнить следующие шаги:
    а. рассчитать среднее число пар в день (если больше 6 – считать субботу рабочим днем) используя общую учебную нагрузку для данной группы;
    б. до тех пор, пока существуют актуальные рабочие нагрузки, заполнять каждый день парами (учитывая среднее число пар в день), начиная с первой (если пары не выставлены), пытаясь объединять остальные группы в потоки, подбирая подходящие аудитории, а в случае невозможности нахождения подходящей аудитории – делить потоки пополам (большую половину групп оставлять) до тех пор, пока подходящая аудитория не будет найдена.

3.3 Интерфейс пользователя

При запуске приложения появляется загрузочное окно, которое автоматически сменяется главным окном программы (см. рисунок 3.1): здесь можно создать, либо открыть имеющееся расписание.

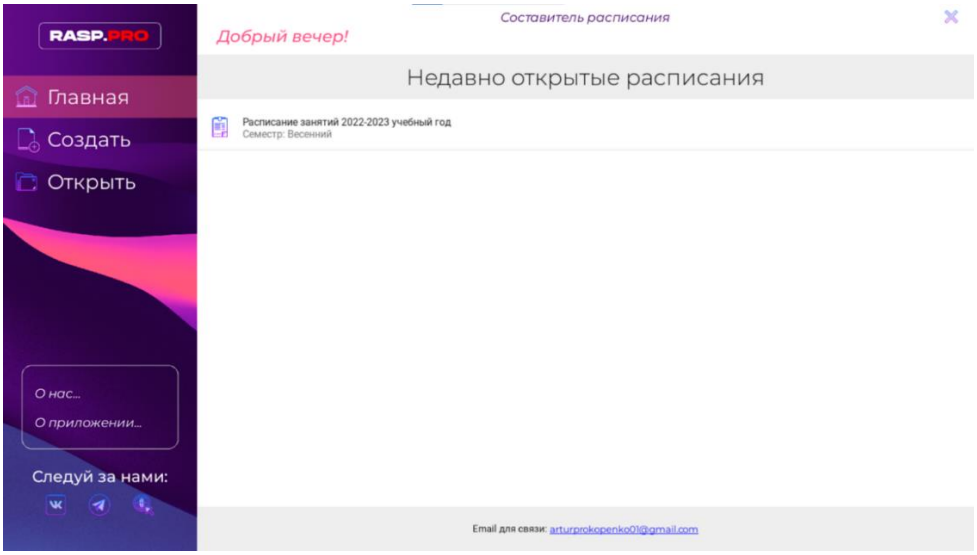

Рисунок 3.1 – Главное окно программы

Открыв расписание, пользователь перенаправляется на окно расписания (см. рисунок 3.2).

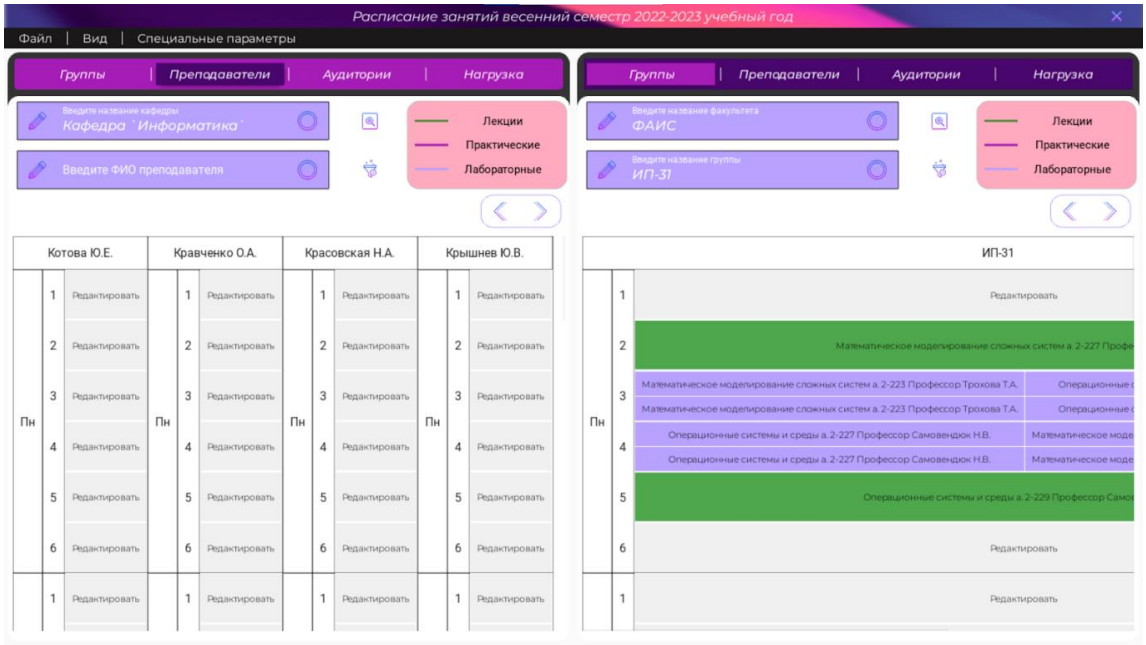

Рисунок 3.2 – Окно расписания

Окно расписания позволяет просматривать расписания в разрезе групп, преподавателей, аудиторий, а также наряды преподавателей. Для удобства поиска добавлены фильтры. На рисунке 3.3 представлено окно расписания с расписанием аудиторий и нарядами преподавателей.

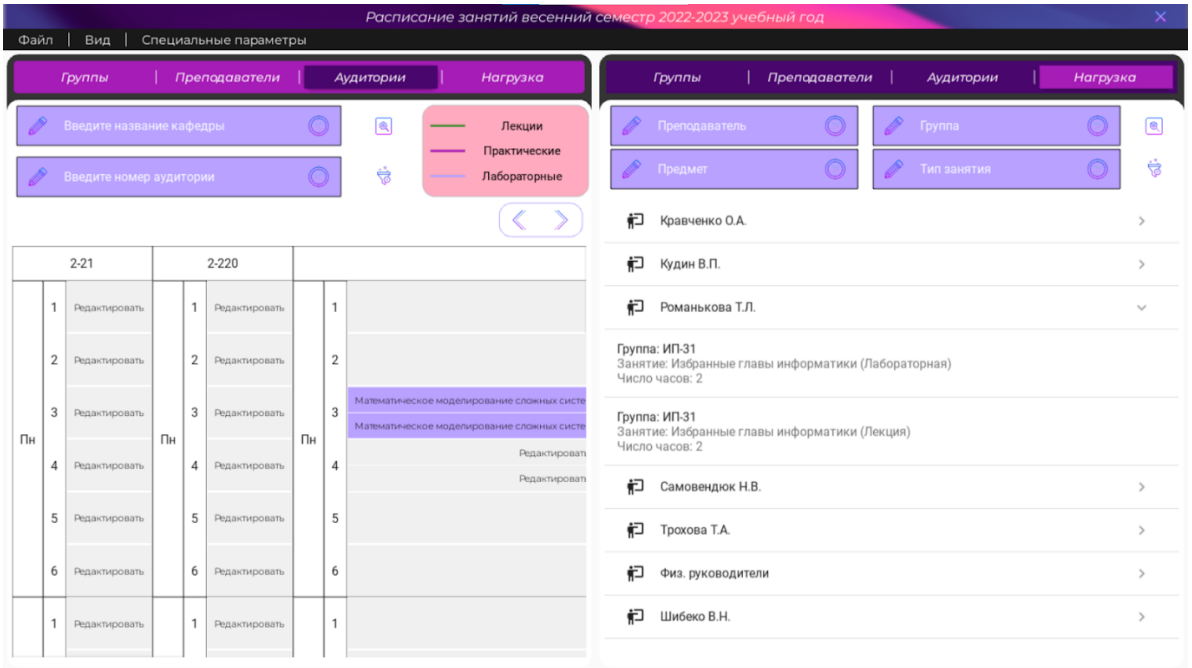

Рисунок 3.3 – Окно расписания с расписанием аудиторий и нарядами преподавателей

Для добавления/удаления записи расписания необходимо нажать левой кнопкой мыши по любой ячейке в любом виде расписания, после чего откроется окно редактирования записи расписания (см. рисунок 3.4).

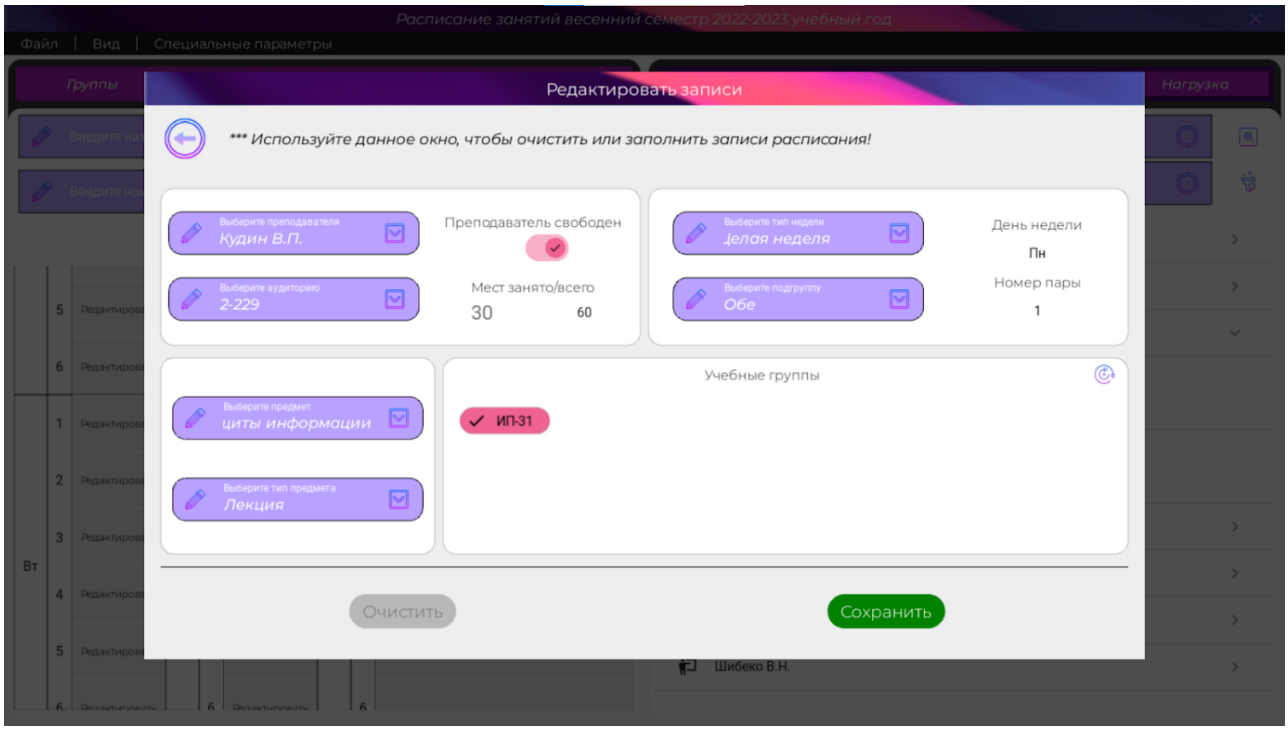

Рисунок 3.4 – Окно редактирования записи расписания

Окно содержит набор пользовательских элементов управления, с помощью которых можно манипулировать текущей ячейкой расписания. Пользовательские элементы управления ограничивают пользователя в выборе, учитывая ограничения системы.

Исходными данными для программы являются справочники базы данных, содержащие сведения о преподавателях, предметах, типах предметов, аудиториях, аудиториях занятий, факультетах, кафедрах, учебных группах, а также нарядах преподавателей.

Выходные данные представлены PDF-документами с расписанием.

4 ТЕСТИРОВАНИЕ, ВЕРИФИКАЦИЯ И ВАЛИДАЦИЯ

4.1 Основные понятия

Тестирование – это выполнение программы с целью обнаружения факта наличия в программе ошибки [22].

Целью тестирования является обнаружение ошибок в ней.

При тестировании приложению передаются входные данные и запрашивается выполнение некой команды, после чего производится проверка полученных результатов на соответствие эталону, если результат соответствует ожидаемому – тест считается пройденным. Эта процедура может быть автоматизирована, в этом случае проверка работоспособности и правильности работы приложения в сравнении с ручным тестированием осуществляется гораздо быстрее, полноценнее и фактически чаще.

Первые программные системы разрабатывались в рамках научных исследований или программ для нужд министерств обороны. Тестирование подобных продуктов проводилось строго формализовано с записью всех тестовых процедур, тестовых данных, полученных результатов. Тестирование выделялось в отдельный процесс, который начинался после завершения кодирования, но при этом, как правило, выполнялось тем же персоналом.

Сегодня же тестированием, как правило, занимаются отдельные специалисты – тестировщики. В их обязанность входит поиск вероятных ошибок и сбоев в функционировании объекта тестирования (продукта, программы). Тестировщик моделирует различные ситуации, которые могут возникнуть в процессе использования предмета тестирования, чтобы разработчики смогли исправить обнаруженные ошибки [23].

В настоящее время существует методология разработки программного обеспечения через тестирование (TDD). Она основывается на повторении коротких циклов разработки: изначально пишется тест, покрывающий желаемое измерение, затем пишется программный код, который реализует желаемое поведение системы и позволит пройти написанный тест, а затем проводится рефакторинг написанного кода с постоянной проверкой прохождения всех тестов.

При написании данной работы использовалась методология разработки ПО через тестирование.

Архитектура программных продуктов, разрабатываемых в соответствии с TDD, обычно лучше, т.к. в приложениях, которые пригодны для автоматического тестирования, обычно очень хорошо распределяется ответственность между компонентами, а выполняемые сложные процедуры декомпозированы на множество простых.

Верификация данных – это процесс проверки данных различных типов по критериям:

– поступление из доверенного источника;

– точность;

– согласованность;

– соответствие формату представления после выполнения операций миграции, трансформации и других операций с данными.

Термин «Верификация» применим как данным, так и их наборам – датасетам. Особую важность верификация данных приобретает в условиях автоматизированной обработки данных в информационных системах, внесении данных в базу данных [24].

Верификация данных позволяет определить, были ли данные точно перенесены из одного источника в другой, являются ли они полными и поддерживают ли процессы в новой системе. В качестве методов верификации можно применять проверку идентичности исходных данных и производных наборов путем побайтового сравнения, подсчета контрольных сумм, вычитки текстов и другие методы.

Верификация данных связана с валидацией данных, но между ними есть существенное отличие. В процессе верификации проверяется формальное соответствие заданным критериям результатов операций с данными, в то время как в процессе валидации проверяется корректность самого набора данных и применимость данных для решения конкретных вычислительных и других задач.

4.2 Ручное тестирование программного обеспечения

При входе в приложение пользователь попадает на главную страницу, откуда можно перейти на сайт разработчика, перейти на страницу «О программе», открыть имеющееся расписание, либо создать новое расписание.

Если пользователь желает создать новое расписание, ему необходимо нажать на кнопку «Создать» (см. рисунок 4.1), после чего ожидается появление диалогового окна, посвященного созданию расписания (см. рисунок 4.2).

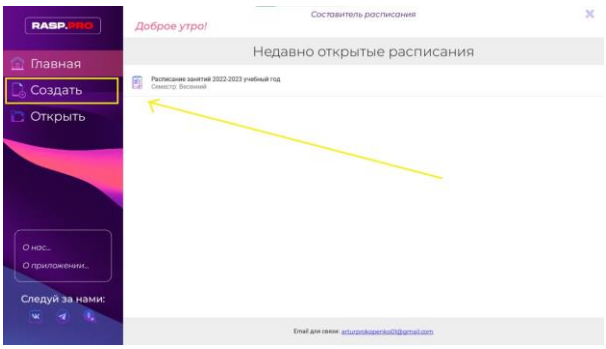

Рисунок 4.1 – Расположение кнопки «Создать» на главном окне

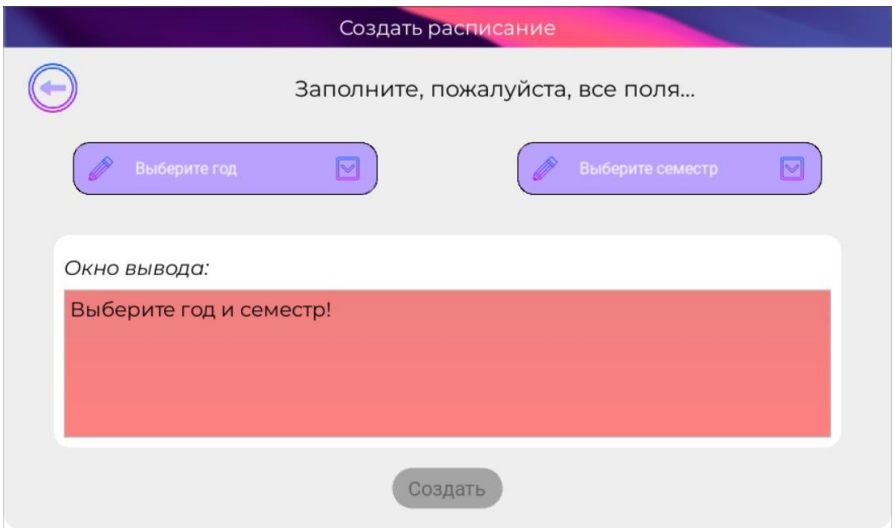

Рисунок 4.2 – Окно создания расписания

Окно, посвященное созданию расписания, содержит два самозаполняющихся поля (используют выпадающие списки для выбора значения) для значений года расписания и семестра. Элементы для выпадающих списков берутся из базы данных и зависят от нагрузок преподавателей. Помимо полей присутствует окно вывода, уведомляющее пользователя о результате создания или причине невозможности создания, а также кнопку создания, активность которой зависит от наполнения полей. Кнопка должна становиться активной после того, как будут заполнены все поля (см рисунок 4.3). После нажатия на нее создается новое расписание (с переходом к окну составления расписания), а если оно уже существует, то об этом пользователь будет уведомлен посредством окна вывода (см. рисунок 4.4).

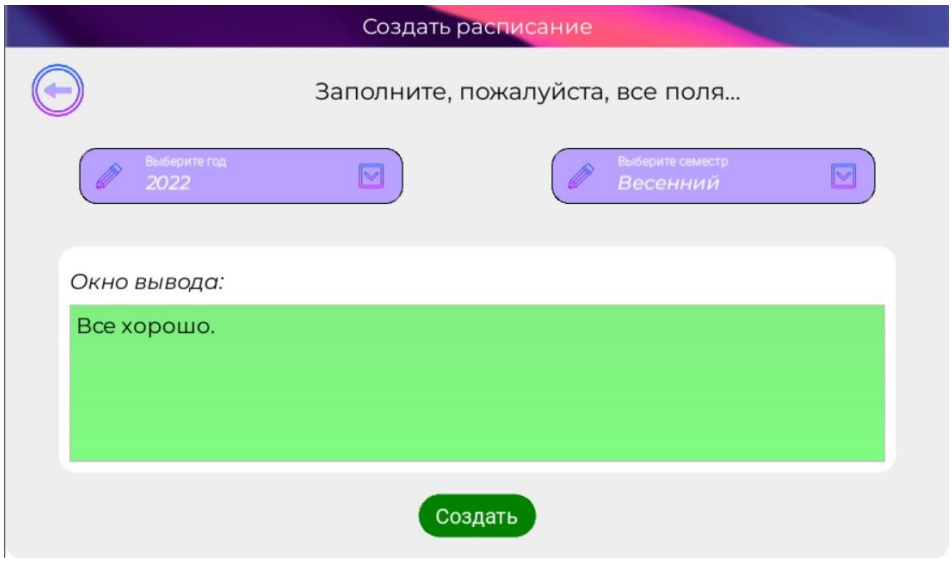

Рисунок 4.3 – Окно создания расписания с заполненными полями и активной кнопкой «Создать»

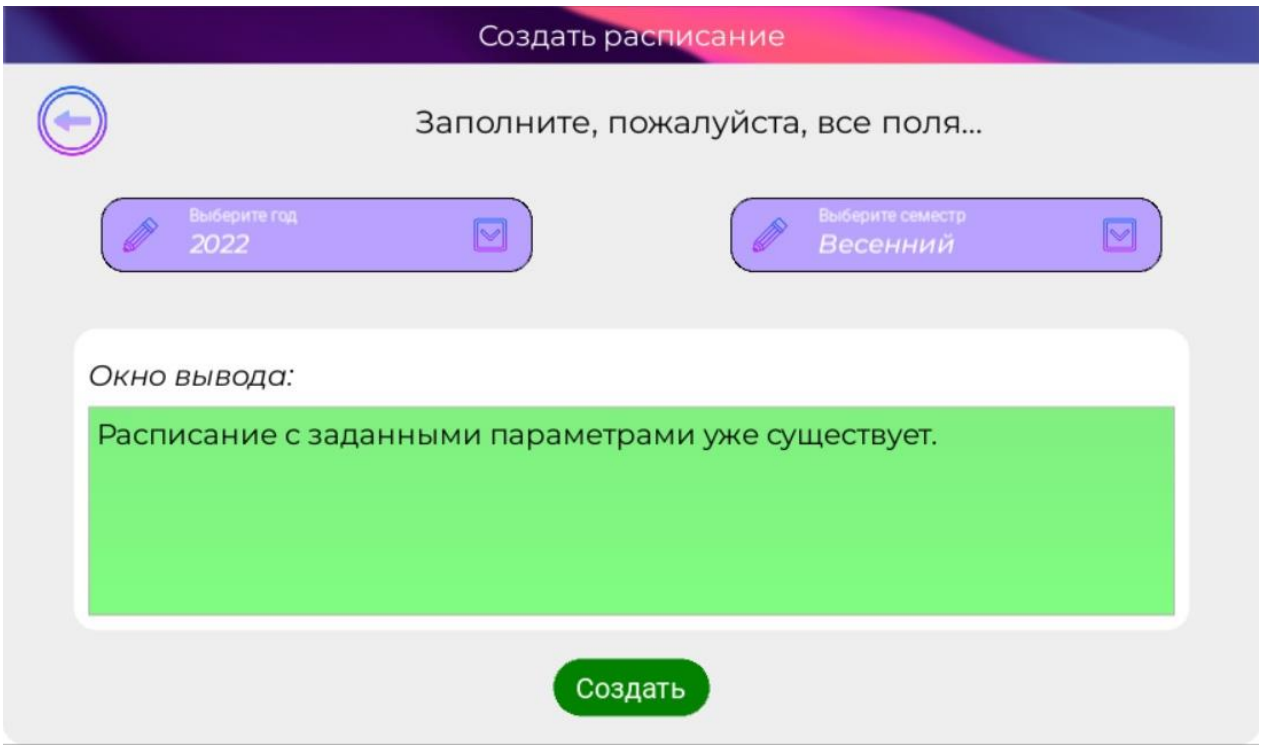

Рисунок 4.4 – Окно создания расписания, уведомляющее пользователя о существовании создаваемого расписания

Для открытия расписания можно воспользоваться списком на главном окне (он состоит из 10 новейших расписаний), а также кнопкой «Открыть» (см. рисунок 4.5). Достаточно кликнуть по элементу списка для открытия расписания.

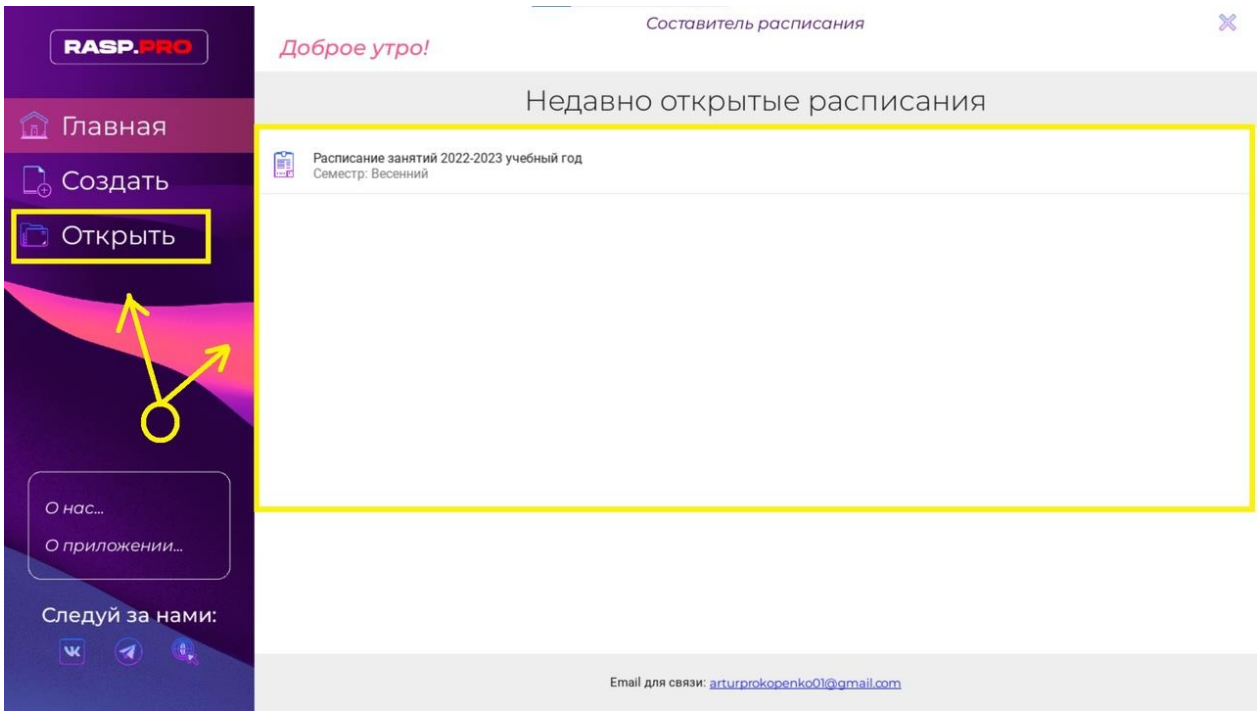

Рисунок 4.5 – Способы открыть расписание

При нажатии на кнопку «Открыть» на главном окне приложения должно появиться диалоговое окно, предназначенное для открытия расписания. Данное окно состоит из панели поиска, кнопки для возвращения на главное окно и списка расписаний (см. рисунок 4.6). По умолчанию список расписаний состоит из всех имеющихся расписаний. Вообще если не указаны год и семестр, то после нажатия на кнопку поиска будут выведены все расписания. Указав год и (или) семестр можно фильтровать расписания для более быстрого поиска. Кнопка «Очистить фильтры» предназначена для установки значений полей по умолчанию.

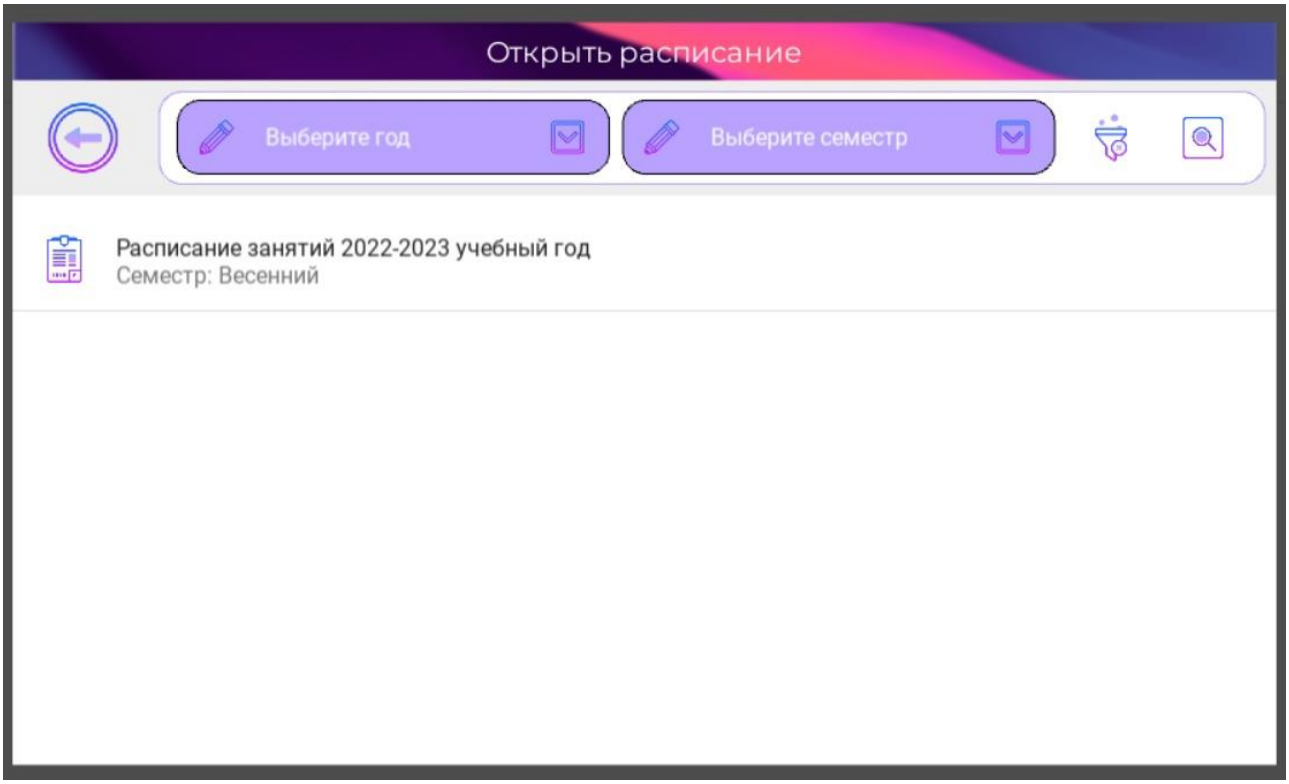

Рисунок 4.6 – Окно открытия расписания

Выбрав интересующее расписание щелчком мыши, пользователь перенаправляется на окно расписания (см. рисунок 3.2). Здесь есть различные вкладки, позволяющие создать, открыть, сохранить, удалить, заполнить автоматически расписание, получить PDF из него, а также вернуться на главное окно. Используя вкладку «Вид», пользователь может настроить пользовательский интерфейс приложения, а вкладка «Специальные параметры» предназначена для указания ответственных за составление расписания лиц, расписания звонков и допустимого числа пар в день (см. рисунок 4.7). Специальные параметры необходимы для генерации PDF-документа с расписанием. Однако, если параметры указаны не будут, подставятся значения по умолчанию.

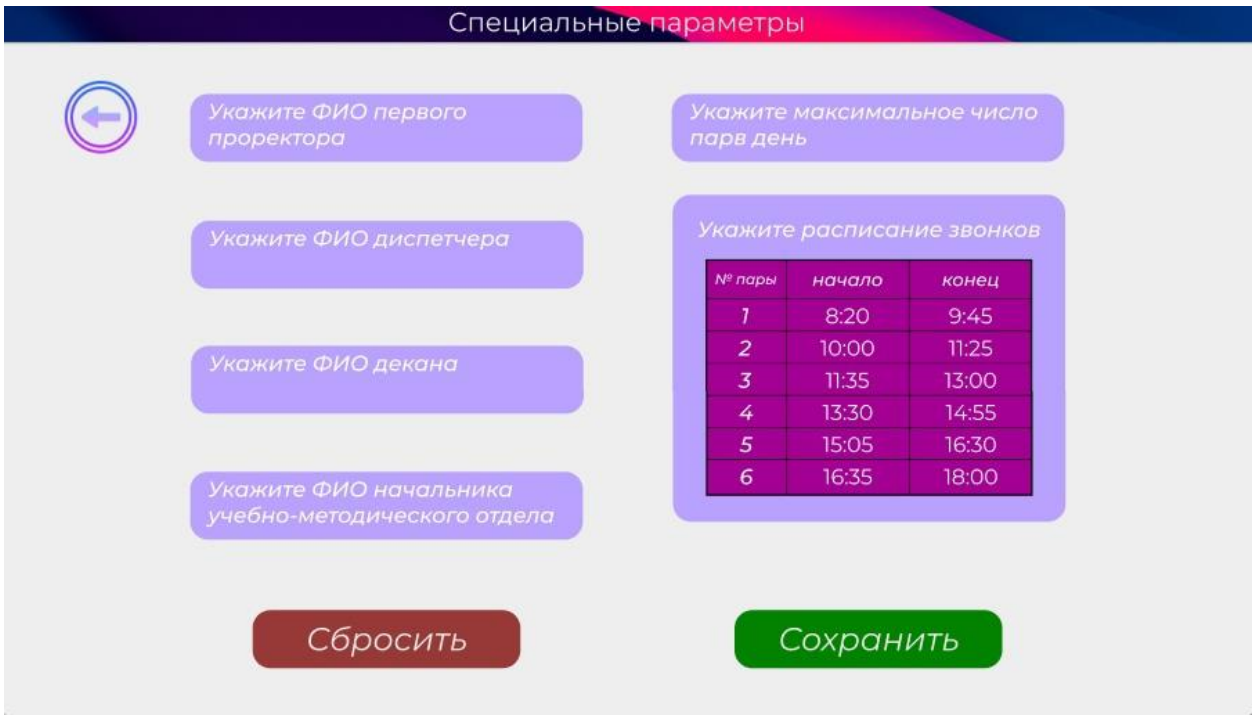

Рисунок 4.7 – Окно специальных параметров

Помимо вкладок, окно расписания содержит две панели выбора вида расписания (просмотра нарядов преподавателей): расписание групп, расписание аудиторий, расписание преподавателей, а также наряды преподавателей. Панели позволяют просматривать расписание сразу в двух любых видах (см. рисунки 3.2, 3.3).

Изначально расписания любых видов не загружены. Число преподавателей, групп и аудиторий в ГГТУ довольно велико и загрузка всех элементов может потребовать немало времени. Вместо загрузки всех элементов лучше использовать фильтры. Элементы, хранящие значения фильтров, позволяют не только выбирать значения из выпадающих списков, но и вводить значения вручную. Поиск ведется по наличию подстроки в строке (например, указав значение «11» для группы, результатом будут все группы с подстрокой «11»: «ИП-11», «ИТ-11» и т.д.) (см. рисунок 4.8). Так, для выбора группы можно указать факультет, к которому она относится, либо название группы. Для преподавателя определяющими фильтрами являются: название кафедры, ФИО преподавателя. Для аудитории фильтры следующие: название кафедры, номер аудитории. И, наконец, наряды преподавателей могут фильтроваться путем указания преподавателя, группы, предмета и типа занятия. Ни один из фильтров не является обязательным. Отсутствие значения для любого фильтра означает то, что подойдет любой из возможных вариантов значений данного фильтра.

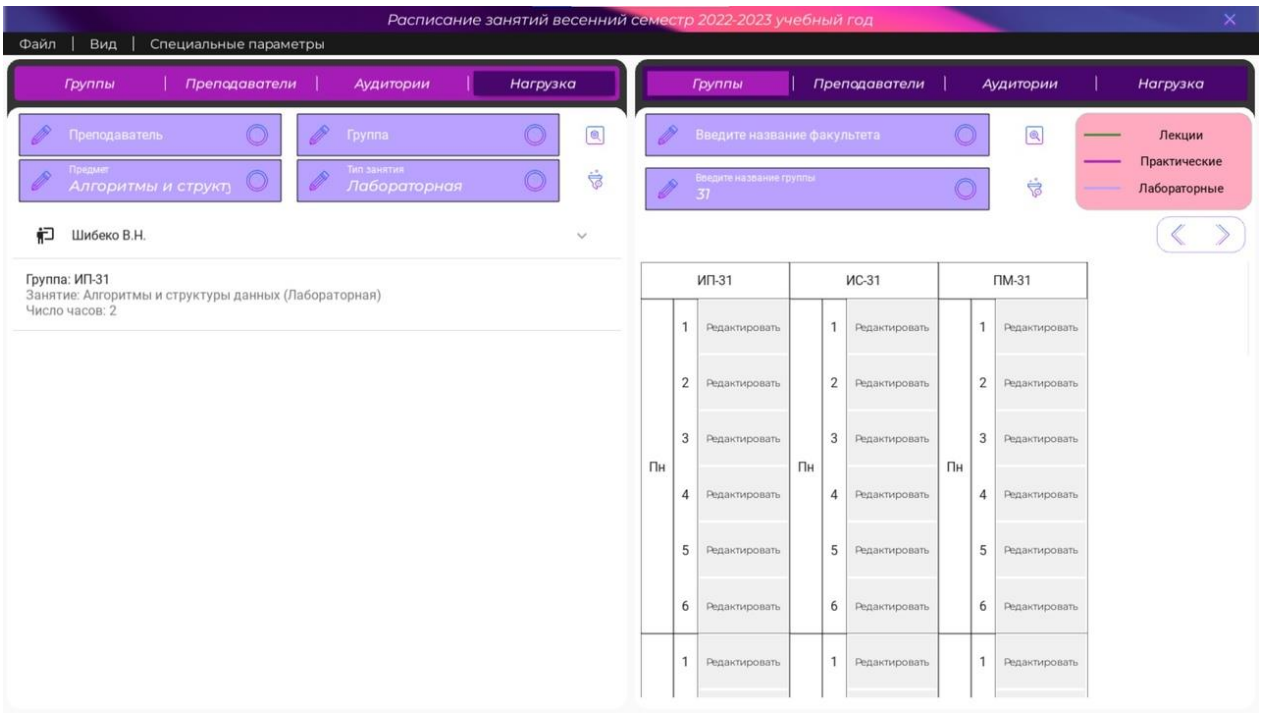

Рисунок 4.8 – Работа фильтров при выборе элементов расписания

Выбрав ячейку расписания, откроется окно редактирования записи расписания (см. рисунок 3.4). Здесь пользователь может выбирать значения только из предлагаемых ему вариантов.

При сохранении или очистке ячейки расписания происходит событие, обновляющее данные всех ячеек всех видов расписаний.

Программный продукт устроен таким образом, что пользователь не может ввести неверные данные. Набор пользовательских элементов управления спроектирован специально для того, чтобы каждый элемент мог обращаться к модели данных через общий контроллер и получать достоверную, согласованную информацию. Это означает, что при условии правильности работы всех методов контроллера и модели данных можно быть уверенным в достоверности данных, отображенных на экране. Валидация и верификация обеспечиваются путем проектирования пользовательских элементов управления и дизайна приложения.

4.3 Автоматизированное тестирование программного продукта

Для автоматизированного тестирования выберем тестирование *Python*-скрипта.

Тестирование проводилось при помощи модуля *pytest*.

Pytest – это основанная на *Python* среда тестирования, которая используется для написания и выполнения тестовых кодов.

Рассмотрим основные преимущества *pytest*:

– *pytest* может выполнять несколько тестов параллельно, что сокращает время выполнения набора тестов;

– у *pytest* есть собственный способ автоматического определения тестового файла и тестовых функций, если это не указано явно;

– *pytest* позволяет нам пропустить подмножество тестов во время выполнения;

– *pytest* позволяет нам запускать подмножество всего набора тестов;

– *pytest* является бесплатным и открытым исходным кодом;

– *pytest* содержит стандартные фикстуры и позволяет легко создавать собственные фикстуры;

– благодаря простому синтаксису, *pytest* очень прост для запуска.

Для запуска тестов необходимо в терминале написать команду «*pytest*». Также запуск можно осуществить из интегрированной среды выполнения «*PyCharm*», кликнув по директории с тестами правой кнопкой и нажав «Запустить, используя *pytest*». Результат запуска тестов представлен на рис. 4.9.

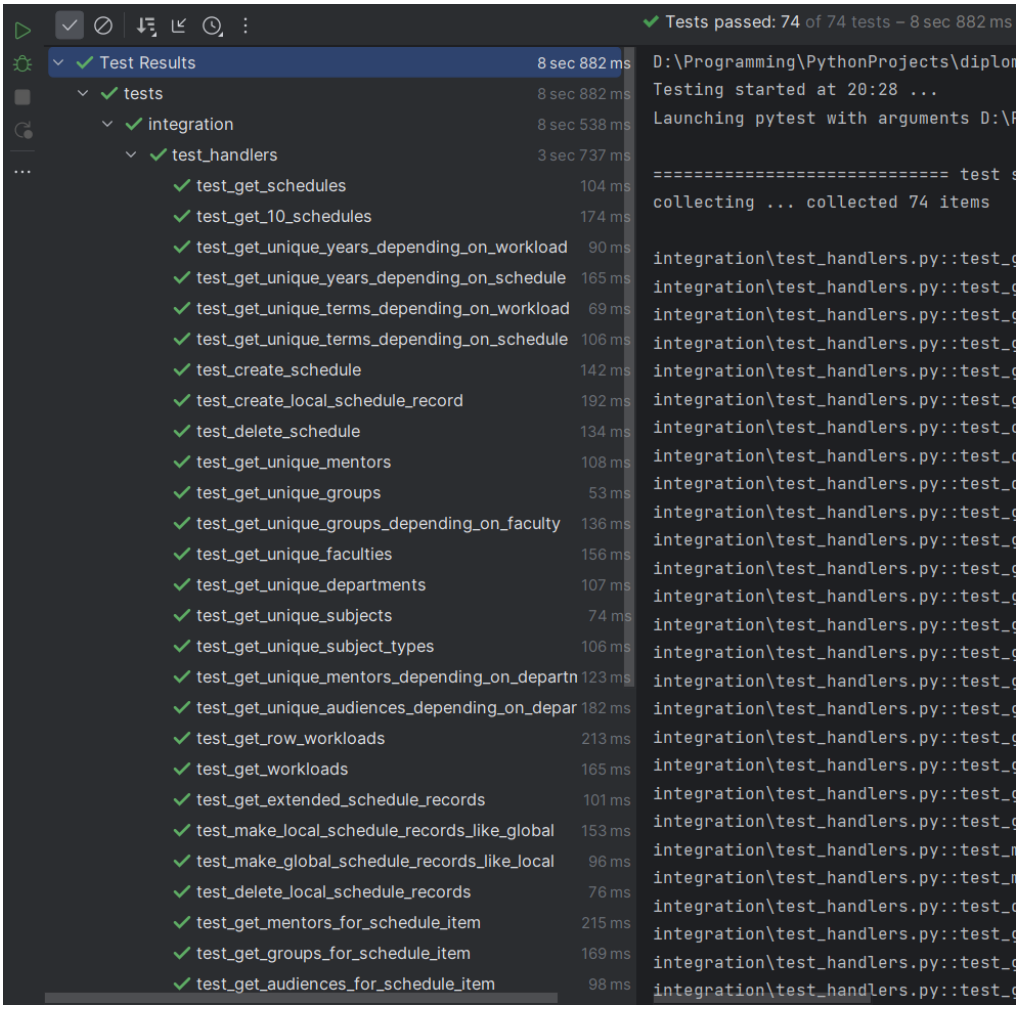

Рисунок 4.9 – Результат запуска автоматизированных тестов

5 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

5.1 Расчет трудоемкости работ по разработке программного обеспечения

Общий объем ПО (V_O) определяется исходя из количества и объема функций, реализуемых программой, по каталогу функций ПО в соответствии с табл. 1.1 в приложении 1 в источнике [21] по формуле (5.1):

$$V_O = \sum_{i=1}^{n} V_i, \quad (5.1)$$

где V_i – объем отдельной функции ПО; n – общее число функций.

В зависимости от организационных и технологических условий, в которых разрабатывается ПО, исполнители по согласованию с руководством организации могут корректировать объем на основе экспертных оценок. Уточненный объем ПО (V_y) определяется по формуле (5.2):

$$V_y = \sum_{i=1}^{n} V_{yi}, \quad (5.2)$$

где V_{yi} – уточненный объем отдельной функции ПО в строках исходного кода (*LOC*).

В таблице 5.1 приведен перечень и объем функций программного обеспечения.

Таблица 5.1 – Перечень и объем функций программного обеспечения

Код функции	Наименование (содержание) функций	Объем функции, строк исходного кода (*LOC*)	
		по каталогу (V_O)	уточненный (V_y)
101	Организация ввода информации	130	20
102	Контроль, предварительная обработка и ввод информации	490	380
109	Управление вводом-выводом	1970	240
201	Генерация структуры базы данных	3500	401
202	Формирование базы данных	1980	9
206	Манипулирование данными	7860	3117

207	Организация поиска и поиск в базе данных	4720	844
208	Реорганизация базы данных	170	18
301	Формирование последовательного файла	590	204
501	Монитор ПО (управление работой компонентов)	1230	2955
506	Обработка ошибочных сбойных ситуаций	1540	9
507	Обеспечение интерфейса между компонентами	1680	359
601	Проведение тестовых испытаний прикладных программ в интерактивном режиме	3780	2403
707	Графический вывод результатов	420	900
	Итого	29470	11859

По своим характеристикам разработанная программа относится ко второй категории сложности программного обеспечения (моделирование объектов и процессов, обеспечение переносимости ПО).

На основании таблицы 1.3 приложения 1 источника [21] была определена нормативная трудоемкость на разработку ПО (табл. 5.2).

Таблица 5.2 – Нормативная трудоемкость на разработку ПО ($T_{\text{н}}$)

Уточненный объем, V_y	2-я категория сложности ПО	Номер нормы
11870	561	65

Дополнительные затраты труда, связанные с повышением сложности разрабатываемого ПО, учитываются посредством коэффициента повышения сложности ПО ($K_{\text{с}}$). $K_{\text{с}}$ рассчитывается по формуле (5.3):

$$K_{\text{с}} = 1 + \sum_{i=1}^{n} K_i, \qquad (5.3)$$

где K_i – коэффициент, соответствующий степени повышения сложности;

n – количество учитываемых характеристик.

$$K_{\text{с}} = 1 + 0{,}06 + 0{,}07 + 0{,}12 = 1{,}25$$

ПО является развитием параметрического ряда ПО, разработанных для ранее освоенных типов конфигурации ПК и ОС, поэтому $K_{\text{н}} = 0{,}63$.

Степень охвата реализуемых функций разрабатываемого ПО стандартными модулями от 20 до 40%, поэтому $K_{\text{т}} = 0{,}77$.

Поскольку используются объектно-ориентированные языки высокого уровня, а также приложение функционирует в Windows, то $K_{\text{у.р}} = 1$.

Поскольку категория новизны ПО – В, $K_{\text{н}} = 0{,}63$ и в ПО не применяются CASE-технологии, то значения коэффициентов удельных весов трудоемкости стадий разработки ПО будут взяты из таблицы 5.3.

Таблица 5.3 – Значения коэффициентов удельных весов трудоемкости стадий разработки ПО

Стадии разработки ПО				
ТЗ	ЭП	ТП	РП	ВН
Значения коэффициентов				
$K_{\text{т.з}}$	$K_{\text{э.п}}$	$K_{\text{т.п}}$	$K_{\text{р.п}}$	$K_{\text{в.н}}$
0,08	0,19	0,28	0,34	0,11

Расчеты распределения нормативной трудоемкости ПО по стадиям, чел.-дн. Отображены в таблице 5.4.

Таблица 5.4 – Расчеты распределения нормативной трудоемкости ПО по стадиям, чел.-дн.

Значения коэффициентов					Нормативная трудоемкость разработки ПО ($T_{\text{н}}$), чел.-дн.
$K_{\text{т.з}}$	$K_{\text{э.п}}$	$K_{\text{т.п}}$	$K_{\text{р.п}}$	$K_{\text{в.н}}$	
0,08	0,19	0,28	0,34	0,11	561 дней
Распределение нормативной трудоемкости ПО по стадиям ($K_{\text{стад}} * T_{\text{н}}$)					
45д.	107д.	157д.	190д.	62д.	== 561 дней

Нормативная трудоемкость ПО ($T_{\text{н}}$) выполняемых работ по стадиям разработки корректируются с учетом коэффициентов: повышения сложности ПО ($K_{\text{с}}$), учитывающих новизну ПО ($K_{\text{н}}$), учитывающих степень использования стандартных модулей ($K_{\text{т}}$), средства разработки ПО ($K_{\text{у.р}}$) и определяются по формулам (5.4-5.8):

для стадии ТЗ

$$T_{\text{у.т.з}} = T_{\text{н}} * K_{\text{т.з}} * K_{\text{с}} * K_{\text{н}} * K_{\text{у.р}}, \quad (5.4)$$

для стадии ЭП

$T_{\text{у.э.п}} = T_{\text{н}} * K_{\text{э.п}} * K_{\text{с}} * K_{\text{н}} * K_{\text{у.р}},$	(5.5)

для стадии ТП

$T_{\text{у.т.п}} = T_{\text{н}} * K_{\text{т.п}} * K_{\text{с}} * K_{\text{н}} * K_{\text{у.р}},$	(5.6)

для стадии РП

$T_{\text{у.р.п}} = T_{\text{н}} * K_{\text{р.п}} * K_{\text{с}} * K_{\text{н}} * K_{\text{т}} * K_{\text{у.р}},$	(5.7)

для стадии ВН

$T_{\text{у.в.н}} = T_{\text{н}} * K_{\text{в.н}} * K_{\text{с}} * K_{\text{н}} * K_{\text{у.р}},$	(5.8)

где $K_{\text{т.з}}$, $K_{\text{э.п}}$, $K_{\text{т.п}}$, $K_{\text{р.п}}$, $K_{\text{в.н}}$ – значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО.

Коэффициенты $K_{\text{с}}$, $K_{\text{н}}$, $K_{\text{у.р}}$ вводятся на всех стадиях разработки, а коэффициент $K_{\text{т}}$ вводится только на стадии РП.

Таким образом, получаем

$$T_{\text{у.т.з}} + T_{\text{у.э.п}} + T_{\text{у.т.п}} = 35{,}43 + 84{,}26 + 123{,}64 \approx 244 \text{ д.}$$

$$T_{\text{у.р.п}} = 115{,}21 \approx 116 \text{ д.}$$

$$T_{\text{у.в.н}} = 48{,}82 \approx 49 \text{ д.}$$

Общая трудоемкость разработки ПО ($T_{\text{о}}$) равна сумме значений нормативной (скорректированной) трудоемкости ПО по стадиям разработки (5.9):

$T_{\text{о}} = \sum_{i=1}^{n} T_{yi},$	(5.9)

где T_{yi} – нормативная (скорректированная) трудоемкость разработки ПО на i-й стадии, чел.-дн.;

n – количество стадий разработки.

Получаем

$$T_{\text{о}} = 244 + 116 + 49 \approx 409 \text{ д.}$$

Результаты расчетов по определению нормативной и скорректированной трудоемкости ПО по стадиям разработки и общей трудоемкости разработки ПО ($T_о$) были внесены в таблицу 5.5.

Таблица 5.5 – Расчет общей трудоемкости разработки ПО

№ п/п	Показатели	Стадии разработки					Итого
		ТЗ	ЭП	ТП	РП	ВН	
1	Общий объем ПО ($V_о$), количество строк LOC	-	-	-	-	-	29470
2	Общий уточненный объем ПО ($V_у$), количество строк LOC	-	-	-	-	-	11859
3	Категория сложности разрабатываемого ПО	-	-	-	-	-	2
4	Нормативная трудоемкость разработки ПО ($T_н$), чел.-дн.	-	-	-	-	-	561д.
5	Коэффициент повышения сложности ПО ($K_с$)	1,25	1,25	1,25	1,25	1,25	-
6	Коэффициент, учитывающий новизну ПО ($K_н$)	0,63	0,63	0,63	0,63	0,63	-
7	Коэффициент, учитывающий степень использования стандартных модулей ($K_т$)	-	-	-	0,77	-	-
8	Коэффициент, учитывающий средства разработки ПО ($K_{у.р}$)	1	1	1	1	1	-
9	Коэффициенты удельных весов трудоемкости стадий разработки ПО ($K_{т.з}$, $K_{э.п}$, $K_{т.п}$, $K_{р.п}$, $K_{в.н}$)	0,08	0,19	0,28	0,34	0,11	1
10	Распределение нормативной трудоемкости ПО по стадиям, чел.-дн.	45д.	107д.	157д.	190д.	62д.	561д.
11	Распределение скорректированной (с учетом $К_с, К_н, К_т, К_{у.р}$) трудоемкости ПО по стадиям, чел.-дн.	36д.	84д.	124д.	116д.	49д.	409д.

В общем виде совокупность капитальных вложений в проект может быть рассчитана следующим образом (5.10):

$K = K_{об} + K_{на} - K_{л} + K_{пр},$	(5.4)

где $K_{об}$ – стоимость устанавливаемого оборудования, руб.;

$K_{на}$ – недоамортизированная часть стоимости демонтируемого оборудования, руб.;

$K_{л}$ – ликвидационная стоимость (выручка от продажи) демонтируемого оборудования, руб.;

$K_{пр}$ – стоимость приобретенных программных продуктов, руб.

$$K = 0 \text{ руб.}$$

Коэффициенты равны нулю, так как оборудование и программные продукты не приобреталось и не демонтировались.

5.2 Расчет затрат на разработку программного продукта

6 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ. ПРОИЗВОДСТВЕННАЯ САНИТАРИЯ И ГИГИЕНА ТРУДА

6.1 Вредные факторы работы на персональных электро-вычислительных машинах (ПЭВМ)

В настоящее время признано, что наибольшему возникновению при работе на ПЭВМ подвергается зрение. Пользователи, как правило, жалуются на покраснение век и глазных яблок, слезотечение, затуманивание зрения, жжение и боли в глазах, боли в области лба. Во многом это связано с особенностями работы с дисплеем ПЭВМ. Изображение на дисплее имеет ряд отличий от печатного текста:

– изображение является самосветящимся и не воспринимается как печатный текст;

– изображение формируется дискретными точками;

– контрастность динамически меняется;

– изображение имеет меняющуюся яркость;

– возникает мерцание изображения;

– имеет место определенная скорость развертки;

– на экране возникают блики от внешних источников света;

– возможно наличие «агрессивных полей» (множество одинаковых элементов – клеток, полос, кругов).

Имеет значение и светотехническая разнородность объектов зрительной работы пользователя (экран, клавиатура, документация). Существенных фактором зрительного утомления является длительность непрерывной работы с дисплеем.

Рабочая поза пользователя ПЭВМ вынужденная, фиксированная с определенным напряжением мышц спины, шеи, верхнего пояса. При большой высоте клавиатуры нарушение естественного угла между кистью и предплечьем вызывает значительное напряжение предплечий и кистей. При выполнении работы с высокой скоростью развивается утомляемость, ощущается дискомфорт и напряжение в спине.

Основную опасность для здоровья пользователей ПЭВМ представляет электромагнитное излучение, создаваемое отклоняющей системой кинескопа видеомонитора, и особенно низкочастотная составляющая ЭМП (до 100 Гц), способствующая изменению биохимической реакции в крови на клеточном уровне. Это приводит к возникновению у человека симптомов раздражительности, нервного напряжения, вызывает осложнения в протекании беременности.

Вместе с этим практически все современные ПЭВМ являются источниками электростатического поля, возникающего в результате облучения экрана

видеомонитора потоком заряженных частиц. Это приводит к накоплению пыли на электростатических экранах, которая воздействует на пользователя во время работы за монитором. Осаждение пыли на поверхности тела может быть причиной кожных заболеваний. Кроме этого, эксперты считают, что низковольтный электромагнитный разряд может способствовать возникновению катаракты, развитию вегетативно-сосудистых расстройств.

Таким образом, повсеместное внедрение ПЭВМ при отсутствии должного контроля за их эксплуатацией и проведении недостаточных мер медицинской профилактики не исключает возможности развития ряда отрицательных состояний у пользователей этих технических устройств. Это, в свою очередь, предопределяет необходимость использования надежной системы защиты и адекватных мер медицинской профилактики с целью обеспечения безопасной работы на компьютерах [19].

6.2 Рекомендации по обеспечению безопасности при работе на ПЭВМ

6.2.1 Гигиеническое нормирование рабочих параметров при эксплуатации ПЭВМ регламентируется санитарно-эпидемиологическими правилами и нормативами СанПиН 2.2.2/2ю4ю1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы». Эти требования направлены на предотвращение неблагоприятного влияния на здоровье человека вредных факторов производственной среды и трудового процесса при работе с ПЭВМ.

Для обеспечения надежного считывания информации при соответствующей степени комфортности ее восприятия визуальные эргономические параметры и пределы их изменений должны соответствовать значениям, приведенным в табл. 6.1. Конструкция ПЭВМ должна обеспечивать мощность эквивалентной дозы рентгеновского излучения на расстоянии 0,5 м от экрана и корпуса не более 0,1 мбэр/ч (100 мкР/ч).

Таблица 6.1 – Визуальные эргономические параметры и пределы их изменений

Параметр	**Пределы значений параметров**	
	Min (не менее)	**Max (не более)**
Яркость знака, яркость фона (измеренная в темноте), кд/м^2	35	120
Внешняя освещенность экрана, лк	100	250
Угловой размер экрана, угл. мин	16	60

Допустимые значения параметров неионизирующих электромагнитных излучений приведены в табл. 6.2.

Таблица 6.2 – Допустимые значения параметров неионизирующих электромагнитных излучений

Параметр	**Допустимое значение**
Напряженность электромагнитного поля по электрической составляющей на расстоянии 50 см от поверхности видеомонитора, В/м	10
Напряженность электромагнитного поля по магнитной составляющей на расстоянии 50 см от поверхности видеомонитора, А/м	0,3
Напряженность электростатического поля не должны превышать, кВ/м:	
для взрослых пользователей	20
для детей дошкольных учреждений и учащихся средних специальных и высших учебных заведений	15
Напряженность электромагнитного поля на расстоянии 50 см вокруг видеодисплейного терминала по электрической составляющей должна быть не более, В/м:	
в диапазоне 5 Гц – 2 кГц	25
в диапазоне частот 2-400 кГц	2,5
Плотность магнитного потока должна быть не более, нТл:	
в диапазоне частот 5 Гц – 2 кГц	250
в диапазоне частот 2-400 кГц	25
Поверхностный электростатический потенциал не должен превышать, В	500

В производственных помещениях, в которых работа на ПЭВМ является вспомогательной, температура, относительная влажность и скорость движения воздуха должны соответствовать действующим санитарным нормам микроклимата производственных помещений. В производственных помещениях, где работа на ПЭВМ является основной (диспетчерские, операторские, расчетные, кабины и посты управления, залы вычислительной техники и др.), должны обеспечиваться оптимальные параметры микроклимата.

Содержание вредных химических веществ в воздухе производственных помещений, в которых работа на ПЭВМ является вспомогательной, не должно превышать предельно допустимых концентраций вредных веществ в воздухе рабочей зоны, а для помещений, где эта работа является основной, - предельно допустимых концентраций загрязняющих веществ в атмосферном воздухе населенных мест.

При выполнении основной работы на ПЭВМ уровень шума на рабочем месте не должен превышать 50 дБА. Регламентирование шума при работе с

ПЭВМ предусмотрено в октавных полосах частот со среднегеометрическими значениями 31,5; 63; 125; 250; 500; 1000; 2000; 4000; 8000 Гц.

Допустимые значения виброскорости и виброускорения в м/с, м/с^2 и дБ установлены для среднегеометрических частот полос 1,6; 2,0; 2,5; 3,15; 4,0; 5,0; 6,3; 8,0; 10,0; 12,5; 16,0; 20,0; 25,0; 31,5; 40,0; 50,0; 63,0; 80,5 Гц.

Помещения с ПЭВМ должны иметь естественное и искусственное освещение. Естественное освещение должно осуществляться через светопроемы и обеспечивать коэффициент естественной освещенности (КЕО) не ниже 1,2% в зонах с устойчивым снежным покровом и не ниже 1,5% на остальной территории.

Искусственное освещение в помещениях эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения, а в случаях преимущественной работы с документами допускается применение системы комбинированного освещения, когда дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов. Освещенность на поверхности стола в зоне размещения рабочего документа должна составлять 300-500 лк. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк. Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окон, светильников и др.), находящихся в поле зрения, должна быть не более 200 кд/м^2. Необходимо ограничивать и отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при котором яркость бликов на экране не должна превышать 40 кд/м^2, а яркость потолка при применении системы отраженного освещения не должна превышать 200 кд/м^2.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ. Соотношение между яркостью рабочих поверхностей не должно превышать 3:1 – 5:1, а между яркостью рабочих поверхностей стен и оборудования – 10:1.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы типа ЛБ. При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп мощностью до 250 Вт. Допускается применение ламп накаливания в светильниках местного освещения. Светильники местного освещения должны иметь непросвечивающий отражатель с защитным углом не менее 40°.

Коэффициент пульсации не должен превышать 5%, что должно обеспечиваться применением газоразрядных ламп с высокочастотными пускорегулирующими аппаратами (ВЧ ПРА) для любых типов светильников.

При отсутствии светильников с ВЧ ПРА лампы многоламповых светильников или рядом расположенные светильники общего освещения следует подключать к разным фазам трехфазной сети.

Для обеспечения нормируемой освещенности в помещениях использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп [19].

6.2.2 Площадь одного рабочего места с ПЭВМ для взрослых пользователей должна составлять не менее 6,0 $м^2$, а объем – не менее 20,0 $м^3$. Рабочие места с ПЭВМ по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева. Расстояния между рабочими столами с видеомониторами должно быть не менее 2,0 $м^2$, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, а также характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) должен выбираться в зависимости от характера и продолжительности работы на ПЭВМ и с учетом роста пользователя. Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья. При этом регулировка каждого параметра должна быть независимой и иметь надежную фиксацию. Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, неэлектризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Помещения с ПЭВМ должны быть снабжены аптечкой первой помощи и углекислотными огнетушителями, ежедневно в них должна проводиться влажная уборка [19].

6.2.3 Режимы труда и отдыха при работе с ПЭВМ организуются в зависимости от вида и категории трудовой деятельности. Виды трудовой деятельности разделяются на три группы: группа А – работа по считыванию информации с экрана с предварительным запросом; группа Б – работа по вводу

информации; группа В – творческая работа в режиме диалога с ЭВМ. При выполнении в течение рабочей смены работ, относящихся к разным видам трудовой деятельности, за основную работу с ПЭВМ следует принимать такую, которая занимает не менее 50% времени в течение рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливаются три категории тяжести и напряженности работ с ПЭВМ, которые определяются: для группы А – по суммарному числу считываемых знаков за рабочую смену, но не более 60 000 знаков за смену (I – до 20 000, II – до 40 000. III – до 60 000); для группы Б – по суммарному числу считываемых или вводимых знаков за рабочую смену, но не более 40 000 знаков за смену (I – до 15 000, II – до 30 000, III – до 40 000); для группы В – по суммарному времени непосредственной работы в ПЭВМ за рабочую смену, но не более 6 часов за смену (I – до 2 ч, II – до 4, III – до 6 ч).

При 8-часовой рабочей смене и работе на ПЭВМ регламентированные перерывы следует устанавливать:

– для I категории работ – через 2 часа от начала рабочей смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;

– для II категории работ – через 2 часа от начала рабочей смены и через 1,5-2 часа после обеденного перерыва продолжительностью 15 минут или продолжительностью 10 минут через каждый час работы;

– для III категории работ – через 1,5-2 часа от начала рабочей смены и через 1,5-2 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

При работе с ПЭВМ в ночную смену (с 22 до 8 часов) независимо от категории и вида трудовой деятельности продолжительность регламентированных перерывов должна увеличиваться на 60 минут.

При 12-часовой рабочей смене регламентированные перерывы должны устанавливаться в первые 8 часов работы аналогично перерывам при 8-часовой рабочей смене, а в течение последних 4 часов работы (независимо от категории и вида трудовой деятельности) – каждый час продолжительностью 15 минут.

Во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии целесообразно выполнять рекомендованные комплексы упражнений.

С целью уменьшения отрицательного влияния монотонии целесообразно изменять содержание выполняемых работ.

Работающим на ПЭВМ с высоким уровнем напряженности во время регламентированных перерывов и в конце рабочего дня показана психологическая разгрузка в специально оборудованных помещениях (комната психологической разгрузки) [19].

6.3 Медицинские осмотры

В системе профилактических мероприятий, направленных на предупреждение профессиональных и производственно обусловленных заболеваний, важное место занимают медицинские осмотры – контрольное медицинское изучение состояния здоровья работников.

Предварительный медицинский осмотр проводится при приеме на работу. Цель медицинского осмотра заключается в определении возможности данного человека по состоянию здоровья трудиться в конкретных условиях или профессии. При этом задача медицинского осмотра – выявление возможных противопоказаний в состоянии здоровья у обследуемого, препятствующих приему на работу. Перечень противопоказаний для конкретных гигиенических условий труда и профессий утверждается Министерством здравоохранения. Кроме того, предварительный медицинский осмотр часто позволяет выявить хронические формы заболеваний, требующих постановки работника на диспансерный учет.

Периодические медицинские осмотры служат цели динамического наблюдения за состоянием здоровья работающих в условиях воздействия вредных профессиональных факторов. Задача осмотра: выявление начальных признаков профессиональных заболеваний, диагностика общих заболеваний, препятствующих продолжению работы по профессии. В то же время определяются профилактические и реабилитационные мероприятия для восстановления нарушенных функций.

Итоги медицинских осмотров оформляются в виде индивидуальных заключений о годности к работе по данной специальности либо о временном или постоянном переводе на другую работу.

В случае установления признаком профессионального заболевания у работника он направляется в центр профпатологии или отделение профпатологии областной, республиканской больницы для окончательной постановки диагноза профессионального заболевания. Все лица с профессиональными заболеваниями или с отклонениями в состоянии здоровья, которые можно связать с профессией, должны находиться на диспансерном учете.

По результатам периодических медицинских осмотром составляются заключительные акты с участием специалиста по гигиене труда и принимаются решения о продолжении или прекращении трудовой деятельности в профессии обследованных, а также о проведении санитарно-технических мероприятий и осуществлении оздоровительных мер [20].

7 РЕСУРСО- И ЭНЕРГОСБЕРЕЖЕНИЕ

7.1

ЗАКЛЮЧЕНИЕ

Список использованных источников

1. Аганина, Д. А. Проблемы автоматизированного расписания образовательного процесса / Д. А. Аганина. // Молодой ученый. – 2018. – № 42 (228). – С. 42-43.
2. Расписание : Свободная энциклопедия. – Электрон. данные. – Режим доступа: https://ru.wikipedia.org/wiki/Расписание#cite_ref-6. Дата доступа: 16.04.2023.
3. Теория расписаний и вычислительные машины / под ред. Э. Г. Коффмана. – М.: Наука, 1984.
4. Конвей, Р. В. Теория расписаний / Р. В. Конвей, В. Л. Максвелл, Л. В. Миллер – М.: Главная редакция физико-математической литературы изд-ва "Наука", 1975.
5. Танаев В.С. Введение в теорию расписаний / В.С. Танаев, В.В. Шкурба – М.: Главная редакция физико-математической литературы изд-ва "Наука", 1975.
6. Фаулер, М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.
7. Трохова, Т. А. Функциональное моделирование программных систем в UML : методические указания по курсу "Технологии проектирования программного обеспечения информационных систем" для слушателей специальности 1-40 01 73 "Программное обеспечение информационных систем" / Т. А. Трохова. – Гомель : ГГТУ им. П. О. Сухого, 2012. – 34 с.
8. Кузьмичев, А. Б. О подходе к автоматизации составления расписания в учебном заведении // Техника машиностроения : журнал. – 2014. – № 3. – С. 23-26.
9. Авторасписание. – Электрон. данные. – Режим доступа: https://www.mmis.ru/programs/avtor. Дата доступа: 16.04.2023.
10. Электронное расписание. – Электрон. данные. – Режим доступа: http://it-institut.ru/#top_content. Дата доступа: 16.04.2023.
11. Альтхофф, Кори. Сам себе программист. Как научиться программировать и устроиться в Ebay? – Пер. с англ. М. А. Райтмана. – М: Эксмо, 2018. – 208 с.
12. Возможности – PyCharm. Электрон. данные. – Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/features/. Дата доступа: 16.04.2023.
13. Лузанов, П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Лёвшин. – 3-е изд. – Минск : ДМК, 2017. – 118 с.
14. Создание приложений для Android, iOS, macOS и Windows на Python. Электрон. данные. – Режим доступа: https://python-scripts.com/kivy-android-ios-exe. Дата доступа: 16.04.2023.
15. Основы контейнеризации (обзор Docker и Podman) : Хабр. Электрон. данные. – Режим доступа: https://habr.com/ru/articles/659049/. Дата доступа: 16.04.2023.

16. Персиваль Г. Паттерны разработки на Python: TDD, DDD и событийно-ориентированная архитектура / Г. Персиваль, Б. Грегори – СПб.: Питер, 2022. – 336 с.: ил. – (Серия «Для профессионалов»).

17. Дзен Пайтона : Свободная энциклопедия. – Электрон. данные. – Режим доступа: https://ru.wikipedia.org/wiki/Дзен_Пайтона. Дата доступа: 15.05.2023.

18. SOLID : Свободная энциклопедия. – Электрон. данные. – Режим доступа: https://ru.wikipedia.org/wiki/SOLID_(программирование). Дата доступа: 15.05.2023.

19. Азизов Б.М. Производственная санитария и гигиена труда: учеб. пособие / Б.М. Азизов, И.В. Чепегин. – М.: ИНФРА-М, 2015. – 432 с.

20. Измеров, Н. Ф. Гигиена труда : учебник / под ред. Н.Ф. Измерова, В.Ф. Кириллова. – М.: ГЭОТАР-Медиа, 2-16. – 480 с.: ил.

21. Кожевников, Е.А. Расчет экономической эффективности разработки программных продуктов : метод. указания по подготовке организац.-экон. раздела дипломных работ для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» днев. Формы обучения / Е.А. Кожевников, Н.В. Ермалинская. – Гомель : ГГТУ им. П.О. Сухого, 2012. – 68 с.

22. Плаксин М.А. Тестирование и отладка программ – для профессионалов будущих и настоящих / М.А. Плаксин. – М.: БИНОМ. Лаборатория знаний, 2007. – 167 с.: ил.

23. Тестировщик : Свободная энциклопедия. – Электрон. данные. – Режим доступа: https://ru.wikipedia.org/wiki/Тестировщик. Дата доступа: 16.04.2023.

24. Верификация данных : Свободная энциклопедия. – Электрон. данные. – Режим доступа: https://ru.wikipedia.org/wiki/Верификация_данных. Дата доступа: 16.04.2023.